# arXiv:2210.07063v1 [cs.LG] 13 Oct 2022

# Deep Clustering With Consensus Representations

Lukas Miklautz<sup>1,2</sup>, Martin Teuffenbach<sup>1,2</sup>, Pascal Weber<sup>1,2</sup>, Rona Perjuci<sup>1</sup>,

Walid Durani<sup>3</sup>, Christian Böhm<sup>1</sup>, Claudia Plant<sup>1,4</sup>

<sup>1</sup>Faculty of Computer Science, University of Vienna, Vienna, Austria

<sup>2</sup>UniVie Doctoral School Computer Science, Vienna, Austria

<sup>3</sup>Institute of Informatics, Ludwig Maximilian University of Munich, Munich, Germany

<sup>4</sup>ds:UniVie, Vienna, Austria

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

Abstract—The field of deep clustering combines deep learning and clustering to learn representations that improve both the learned representation and the performance of the considered clustering method. Most existing deep clustering methods are designed for a single clustering method, e.g., k-means, spectral clustering, or Gaussian mixture models, but it is well known that no clustering algorithm works best in all circumstances. Consensus clustering tries to alleviate the individual weaknesses of clustering algorithms by building a consensus between members of a clustering ensemble. Currently, there is no deep clustering method that can include multiple heterogeneous clustering algorithms in an ensemble to update representations and clusterings together. To close this gap, we introduce the idea of a consensus representation that maximizes the agreement between ensemble members. Further, we propose DECCS (Deep Embedded Clustering with Consensus representationS), a deep consensus clustering method that learns a consensus representation by enhancing the embedded space to such a degree that all ensemble members agree on a common clustering result. Our contributions are the following: (1) We introduce the idea of learning consensus representations for heterogeneous clusterings, a novel notion to approach consensus clustering. (2) We propose DECCS, the first deep clustering method that jointly improves the representation and clustering results of multiple heterogeneous clustering algorithms. (3) We show in experiments that learning a consensus representation with DECCS is outperforming several relevant baselines from deep clustering and consensus clustering.

Index Terms—Deep Clustering, Representation Learning, Consensus Clustering

### I. INTRODUCTION

Clustering is the task of unsupervised classification, where we infer cluster labels from the data.<sup>1</sup> Deep clustering (DC) combines unsupervised deep learning and clustering to learn representations (embeddings) that improve clustering performance. Current DC methods are designed with only a single clustering model in mind, e.g., DEC [1] which improves the representation for *k*-means [2], VaDE [3] for Gaussian mixture models [4], DeepECT [5] for hierarchical clustering [6], and SpectralNet [7] for spectral clustering [8]. Relying on the assumptions of a single clustering model leads to poor results if the assumptions are not met by the data.

Consensus clustering (CC) can alleviate the limitations of individual clusterings by combining a clustering ensemble into a single robust clustering [9]. Unfortunately, applying

<sup>1</sup>Our code is available at https://gitlab.cs.univie.ac.at/lukas/deccs.

(a) Original (b) Initial AE (c) Update (d) Final CR

Fig. 1. A synthetic data set (a) containing four clusters is embedded (b) with an autoencoder (AE). DECCS transforms the initial AE embedding via several updates (c) to the final consensus representation (CR) in which clusters are compact and well separated (d).

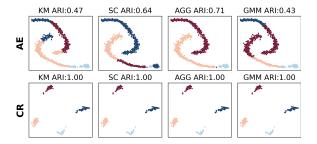


Fig. 2. Cluster performance on initial AE embedding and learned CR for an ensemble of k-means (KM), Spectral clustering (SC), Agglomerative clustering (AGG), and Gaussian mixture model (GMM).

current CC methods to high-dimensional data sets leads to unsatisfactory results, because they are either limited to linear transformations [10]–[12], only work for k-means like clusterings [13], or only use CC information as input features for DC without updating the CC in response to improved data representations [14], [15].

In contrast to that, we propose our novel Deep Embedded Clustering with Consensus representationS (DECCS) method, which is a DC method that can be applied to high-dimensional data, finds non-linearly hidden clusters and works with many existing clustering algorithms. DECCS learns a *consensus representation* (CR) that maximizes the agreement between ensemble members. The key idea we use for consensus representation learning with DECCS is that most clustering methods can find well-separated clusters in a low-dimensional space that have a simple shape, e.g., dense, spherical clusters. Using this idea, DECCS learns a consensus representation by transforming the embedded space such that it is trivial to cluster and, therefore, all ensemble members naturally agree on one partitioning into clusters. Fig. 1 illustrates on a synthetic data

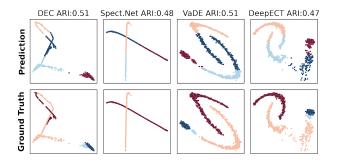


Fig. 3. Existing DC methods are limited by their assumed cluster model. Here they fail, because the data contains clusters of differing shapes.

set how DECCS transforms an initial embedding that contains clusters of different shapes to a consensus representation that consists only of dense, spherical, and well separated clusters. In Fig. 2, multiple, heterogeneous algorithms with different assumptions about the cluster structure are applied to the initial autoencoder (AE) embedding (upper row). Initially, the clustering algorithms perform poorly, but applied to the consensus representation learned with DECCS all algorithms in the ensemble reach the same, perfect clustering (bottom row) as measured with the adjusted rand index (ARI) [16]. In Fig. 3, we apply existing DC methods to the same synthetic data set using the same AE and plot their learned embeddings. While this data set can be clustered by classical clustering techniques, we see that DC methods fail, because their assumptions are not met. For example, DEC is performing poorly because the data set contains non-spherical clusters, which is not suited for k-means. As a consequence DEC is producing a distorted embedding (first column).

In this work, we tackle the shortcomings of existing DC and CC techniques and present the following contributions:

- (1) We introduce the idea of a consensus representation, which is a representation that maximizes the agreement of the applied clustering algorithms by producing similar clustering results for all clustering methods included in the ensemble.
- (2) We propose DECCS, the first DC algorithm that can include multiple heterogeneous clustering methods to jointly improve the learned embedding and clustering results by simplifying the representation.
- (3) Our method is outperforming several relevant baselines in terms of cluster performance.

### II. BACKGROUND - CONSENSUS CLUSTERING

CC can overcome the limitations of individual clusterings by combining multiple clustering solutions into a single robust partitioning [9]. In general, CC algorithms consist of two stages:

- Generate a set of base partitions using single clustering algorithms (e.g., k-means, Spectral Clustering, etc.)
- 2) Combine the base partitions using a consensus function to obtain a final partition.

Traditionally, the two stages are independent of each other. The consensus function does not access the original features of the data set to find the optimal combination of base partitions.

During the design of the consensus function the goal is to combine a set  $\Pi$  of  $|\Pi|$  partitions  $\pi_i$  into one final clustering  $\pi_{cc}$ , such that  $\pi_{cc}$  agrees as much as possible with the base partitions. In their framework, [9] suggested to use the average pairwise normalized mutual information (ANMI) between the CC and the base clusterings as an objective function to measure the agreement:

$$\pi_{cc} = \operatorname{argmax}_{\bar{\pi}} \sum_{i=1}^{|\Pi|} \operatorname{NMI}(\pi_i, \bar{\pi})$$
(1)

Using the normalized mutual information (NMI) has the benefit that it is invariant to the permutation and absolute values of cluster labels and allows for a different number of clusters  $k_i$  in each partition  $\pi_i$ . Further, the NMI is symmetric and is 1 if two clusterings match perfectly and 0 if they are independent of each other.

1

Instead of the need to design a consensus function to optimize Eq. 1 our DECCS algorithm learns a (non-linear) consensus function to learn the *consensus representation* as we explain in the following.

# III. OBJECTIVE FUNCTION FOR CONSENSUS REPRESENTATION LEARNING

For our novel problem setting, we use an encoder  $\operatorname{enc}_{\Theta}$  that maps a data point  $\mathbf{x} \in \mathbb{R}^D$  to a typically lower-dimensional embedded vector  $\mathbf{z} \in \mathbb{R}^d$ , where  $\Theta$  are the learnable parameters of the encoder. Then, let  $\mathbf{X}$  be an  $N \times D$  dimensional input data matrix and  $\mathbf{Z} = \operatorname{enc}_{\Theta}(\mathbf{X})$  be an  $N \times d$  dimensional embedded data matrix with d < D. Further, let  $\mathcal{E}$  be a set of heterogeneous clustering algorithms with potentially different number of clusters  $k_i$ , where each  $i^{\text{th}}$  member  $e_i$  produces a clustering result  $\pi_i = e_i(\mathbf{Z})$ . We define the consensus representation in the following.

**Definition 1** (Consensus representation  $\mathbf{Z}_{cr}$ ). Let  $\Theta$ ,  $\operatorname{enc}_{\Theta}$ ,  $\mathbf{X}$ ,  $\mathbf{Z}$ , and  $\mathcal{E}$  be defined as above. The *consensus representation*  $\mathbf{Z}_{cr}$  maximizes the following objective function:

$$f_{\Theta} = c \sum_{i=1}^{|\mathcal{E}|} \sum_{j>i}^{|\mathcal{E}|} \mathrm{NMI}(e_i(\mathrm{enc}_{\Theta}(\mathbf{X})), e_j(\mathrm{enc}_{\Theta}(\mathbf{X}))), \quad (2)$$

with  $\mathbf{Z}_{cr} := \operatorname{enc}_{\Theta_{cr}}(\mathbf{X})$ , where  $\operatorname{enc}_{\Theta_{cr}}$  is the consensus representation function and c is a normalization constant  $c = \frac{2}{|\mathcal{E}|^2 - |\mathcal{E}|}$  for the equation to sum to one.

The consensus representation maximizes the agreement of all partitions with each other, where the agreement is measured using the pairwise NMI [9]. The optimal encoder parameters for the consensus representation  $\mathbf{Z}_{cr}$  are then learned with

$$\Theta_{cr} = \operatorname{argmax}_{\Theta} f_{\Theta}.$$
(3)

Note that Eq. 2 allows for degenerate solutions, like setting  $\mathbf{Z}_{cr}$  to a constant if  $enc_{\Theta}$  is non-linear. To avoid degenerate

solutions in practice we include regularizers in the objective, like enforcing the invertibility of  $\mathbf{Z}_{cr}$  back to  $\mathbf{X}$  by using the AE reconstruction loss. In the following, we introduce our DECCS method and illustrate how it approaches the consensus representation learning problem.

# **IV. METHOD - DECCS**

We motivate our consensus representation learning approach using the observation that most clustering methods are able to detect compact and well-separated clusters in low-dimensional spaces. DECCS uses the cluster information from all ensemble members to learn such a simplified representation. Decreasing the ambiguity of the representation during training will increase the similarity of the clusterings that ensemble members will produce, which subsequently increases the pairwise NMI (Eq. 2) of clustering results. DECCS works by alternating between representation and clustering update steps until an agreement is reached through the consensus representation. In the following, we explain our approach in more detail.

### A. Overview

We use a (non-linear) autoencoder (AE) to learn enc by reconstructing the original input data x from z using the decoder dec resulting in  $\hat{\mathbf{x}} := \det(\operatorname{enc}(\mathbf{x}))$ .<sup>2</sup> The AE reconstruction  $\hat{\mathbf{x}}$ is learned by minimizing a reconstruction loss  $\mathcal{L}_{rec} = ||\mathbf{x} - \hat{\mathbf{x}}||$ , e.g., using the mean squared error. Given the AE, our DECCS algorithm consists of three main steps that we explain in the following sections. First, we illustrate how we generate a set of base partitions by applying the cluster ensemble to a subsample of the embedding in Section IV-B. Second, we show how to approximate each partition with a classifier to label the remaining data points in Section IV-C. Third, we state our consensus objective in Section IV-D and in Section IV-E, we show how the consensus representation is updated. The algorithm is presented in Section IV-F.

### B. Generating base partitions

At the beginning of each round t of our algorithm, we draw a small random sample  $\mathbf{X}_t$  of size n < N from  $\mathbf{X}$ , because some clustering algorithms are impractical to be applied to large data sets and re-sampling can make the CC more robust [17]. Next, we embed the sample using  $\operatorname{enc}_{\Theta}(\mathbf{X}_t) = \mathbf{Z}_t$  and generate a set of base partitions  $\Pi_t$  by applying all ensemble members to the embedding  $\pi_i = e_i(\mathbf{Z}_t)$ . The sampling procedure and the low-dimensional embedded space allow us to use more run-time and memory expensive algorithms, such as spectral clustering, in our ensembles. Further, using the sampling and heterogeneous ensembles, we can achieve a sufficiently diverse set of base partitions  $\Pi_t$ .

# C. Approximating base partitions

Since we only have cluster labels for n < N data points due to the random sampling, but require cluster labels for all N data points, we use a classifier to approximate the clustering for the remaining N-n points. We approximate the set of base partitions  $\Pi_t$  using a set of classifiers  $\mathcal{G}_t$ , where classifier  $g_i$ is trained to predict the corresponding clustering  $\pi_i$ . We train each classifier by minimizing the cross-entropy loss of cluster labels  $\pi_i$  and its prediction, i.e.,

$$\mathcal{L}_{\rm CE} = \sum_{i=1}^{|\Pi_t|} \mathcal{L}_{CE}^i = -\frac{1}{n} \sum_{j=1}^n \sum_{l=1}^{k_i} I[l = \pi_{i,j}] \log g_i(\mathbf{z}_j), \quad (4)$$

where  $\pi_{i,j}$  is the cluster label corresponding to the  $j^{\text{th}}$  data point and I is the indicator function. While in principle one can use any classifier for  $g_i$  we chose linear classifiers with the Softmax function as output, i.e.,  $g_i(\mathbf{x}) = \operatorname{softmax}(\mathbf{W}_i\mathbf{x} + \mathbf{b}_i)$ with  $\mathbf{W}_i$  and  $\mathbf{b}_i$  as weights and bias terms respectively. The linear classifiers can be trained with little overhead, having only  $d \cdot k_i + k_i$  trainable parameters. Updating the linear classifiers together with the non-linear encoder allows us then to approximate non-linear clusterings as well.

### D. Consensus Objective

Optimizing Eq. 2 from Definition 1 directly is not possible, because the cluster ensemble members are not differentiable. Thus, we learn a low-dimensional representation in which all clusters are spherical, dense, and well separated, such that the ensemble members trivially agree on one partition. To transform non-spherical-shaped clusters into spherical clusters we "move" cluster points closer to their cluster representatives. We choose the mean center of a cluster as the representative because it is stable across update steps, but other choices like the median are also possible. In the following, we use the terms representative and center interchangeably.

Let  $C_{\pi_i}^j$  be the set of data points in the  $j^{\text{th}}$  cluster of partition  $\pi_i$ , then we can calculate its center  $\mu_j$  using  $\mu_j = \frac{1}{|C_{\pi_i}^j|} \sum_{\mathbf{x} \in C_{\pi_i}^j} \operatorname{enc}_{\Theta}(\mathbf{x})$ , and subsequently can construct the  $k_i \times d$  matrix  $M_i$  containing  $k_i$  centers  $\mu_j$  as row vectors. Next, we define our differentiable consensus objective as

$$\mathcal{L}_{\text{cons}} = \sum_{i=1}^{|\Pi_t|} \mathcal{L}_{\text{cons}}^i = \sum_{i=1}^{|\Pi_t|} \|\mathbf{A}_i \mathbf{M}_i - \text{enc}_{\Theta}(\mathbf{X}_t)\|_F^2 \qquad (5)$$
$$= \sum_{i=1}^{|\Pi_t|} \sum_{l=1}^{k_i} \sum_{\mathbf{x} \in \mathcal{C}_{\pi_i}^l} \|\mu_l - \text{enc}_{\Theta}(\mathbf{x})\|_2^2,$$

where  $\mathbf{A}_i$  is the  $n \times k_i$  one hot encoded cluster assignment matrix of partition  $\pi_i$ ,  $\|\cdot\|_2^2$  the squared Euclidean norm, and  $\|\cdot\|_F^2$  the squared Frobenius norm. Here  $\mathbf{M}_i$  and  $\mathbf{A}_i$  are fixed, so the encoder  $\operatorname{enc}_{\Theta}$  has to learn parameters  $\Theta$  that map embedded data points  $\mathbf{z} = \operatorname{enc}_{\Theta}(\mathbf{x})$  as close as possible to their assigned centers across all partitions. Note, that data points that are close to similar centers across partitions will receive a higher gradient update due to the summation and are thus gathering faster than data points that have conflicting assignments. The centers alone can not capture complex cluster structures properly, which is why we include the crossentropy loss (Eq. 4) in our objective, as we explain in the next Section.

 $<sup>^{2}</sup>$ We reuse enc here, whether it is vector or matrix-valued should be clear from the context.

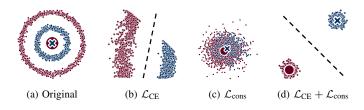


Fig. 4. Example of how optimizing  $\mathcal{L}_{CE}$  together with  $\mathcal{L}_{cons}$  is able to enhance the representation even if cluster centers overlap. (a) A data set with two clusters (red and blue circle) that have the same centers (red dot and blue cross). (b) Optimizing DECCS only with  $\mathcal{L}_{CE}$  learns a representation in which both clusters are separated by the decision boundary (dashed black line), but cluster shapes are distorted. (c) Only using  $\mathcal{L}_{cons}$  merges both clusters. (d) Optimizing  $\mathcal{L}_{CE}$  together with  $\mathcal{L}_{cons}$  separates the clusters and transforms them from circles to spheres, a representation that can be easily clustered.

### E. Updating consensus representation

Putting everything together the objective of our DECCS algorithm is

$$\mathcal{L} = \sum_{i}^{|\Pi_{t}|} \lambda_{i} (\mathcal{L}_{CE}^{i} + \mathcal{L}_{cons}^{i}) + \lambda_{rec} \mathcal{L}_{rec}, \qquad (6)$$

where  $\lambda_i = \frac{1}{|\Pi_t|-1} \sum_{j=1}^{|\Pi_t|} \text{NMI}(\pi_i, \pi_j)$  is a weighting parameter based on the agreement of partition  $\pi_i$  with all other partitions measured as average pairwise NMI to exclude random partitions and downscale outlier partitions. We choose the commonly used AE reconstruction loss  $\mathcal{L}_{rec}$  as data dependent regularizer to avoid degenerate solutions by keeping **Z** approximately invertible. The hyperparameter  $\lambda_{rec}$  weights the importance of  $\mathcal{L}_{rec}$ .

The cross-entropy loss  $\mathcal{L}_{CE}^{i}$  of each classifier is included to make sure that the updated representation is still predictive for each partition, e.g., by avoiding the merging of clusters if centers of different clusters are the same, as illustrated in Fig. 4. Additionally, we show in the Appendix in Fig. 9 that this still works even if half of the ensemble members are no better than chance.

Eq. 6 can be optimized using stochastic gradient descent for a fixed amount of update steps. Once the training has stopped, our algorithm starts again with a new round by applying the clustering algorithms, which will adjust their clustering results to the updated embedding. Then it again pretrains classifiers to approximate them and minimizes  $\mathcal{L}$ . These steps are repeated for several rounds until a stable agreement is achieved or a maximum number of rounds T has been reached. In the following section, we explain our algorithm in more detail.

### F. Algorithm

Given a data set X, the pretrained encoder  $enc_{\Theta}$ , and a parameterized ensemble of clustering methods  $\mathcal{E}$ , we learn a consensus representation  $\mathbf{Z}_{cr}$  and subsequently a consensus clustering  $\pi_{cc}$  with our DECCS algorithm in the following way. We encode the input data using  $enc_{\Theta}$ , generate the base partitions by applying cluster ensemble members on  $\mathbf{Z}_t$ , approximate the base partitions using classifiers  $g_i$  and update the representation by minimizing  $\mathcal{L}$ . We repeat these steps

# Algorithm 1: DECCS

<b>8 • • • • • • • • • •</b>	
<ul> <li>Param : Agreement function a(·), agreement threshold τ, subsample size n, regularization weight λ<sub>rec</sub>, rampup function w(·), maximum number of rounds T, number of mini-batch training iterations ITER</li> <li>Input : data set X initial representation function enc<sub>Θ</sub> ensemble of clustering algorithms E</li> </ul>	_
<b>Output:</b> Estimated consensus representation $\hat{\mathbf{Z}}_{cr}$ and corresponding consensus clustering $\hat{\pi}_{cc}$	
1 $t = 0;$	
2 while $t \leq T$ do	
3 draw sample $\mathbf{X}_t$ of size <i>n</i> from $\mathbf{X}$ and create	
corresponding embedding $\mathbf{Z}_t = \operatorname{enc}_{\Theta}(\mathbf{X}_t);$	
// Generate new base partitions	
4 generate new empty list for cluster partitions $\Pi_t$ ; 5 foreach ensemble member $e_i \in \mathcal{E}$ do	
6   insert cluster prediction $\pi_i = e_i(\mathbf{Z}_t)$ into $\Pi_t$ ;	
// Approximate base partitions	
7 initialize list of classifiers $\mathcal{G}_t$ ;	
s foreach cluster prediction $\pi_i \in \Pi_t$ do	
9 pretrain classifier $g_i$ by minimizing	
$\mathcal{L}_{CE}^{i}(g_{i}(\mathbf{Z}_{t}),\pi_{i});$	
10 insert pretrained classifier $q_i$ into $\mathcal{G}_t$ ;	
// Stop if stable agreement or $T$ is	
reached	
11 <b>if</b> $(t > 0) \land ((  a(\Pi_t) - a(\Pi_{t-1})  _1 < \tau) \lor (t == T))$	
then	
12 $\hat{\mathbf{Z}}_{cr} := \operatorname{enc}_{\Theta}(\mathbf{X}), \ \hat{\pi}_{cc} := g_0(\hat{\mathbf{Z}}_{cr});$	
13 break;	
14 else	
<pre>// Update consensus representation</pre>	
15 while $j < ITER$ do	
16 foreach mini-batch $\mathcal{B} \in X$ do	
17 calculate for $\mathcal{B}$ loss $\mathcal{L}$ :	
18 $\sum_{i}^{ \Pi_t } \lambda_i (\mathcal{L}_{CE}^i + w(t)\mathcal{L}_{cons}^i) + \lambda_{rec}\mathcal{L}_{rec};$	
19 update $enc_{\Theta}$ and $\mathcal{G}_t$ using $\mathcal{L}$ ;	
20 $j = j + 1;$	
21 $t = t + 1;$	
22 return $\hat{\mathbf{Z}}_{cr}, \hat{\pi}_{cc};$	

for several rounds until a stable agreement is achieved or we reached a maximum number of rounds T. As agreement function  $a(\Pi_t)$  we use the average pairwise NMI between all clusterings in the set of partitions  $\Pi_t$ . We measure the stability of the agreement by calculating  $||a(\Pi_t) - a(\Pi_{t-1})||_1 < \tau$ , where  $\tau$  is the cluster agreement threshold, a user-specified parameter, and  $\|\cdot\|_1$  is the absolute distance between the agreement of two subsequent sets of partitions. After the algorithm stops, it returns the estimated consensus representation  $\mathbf{Z}_{cr}$  and it's corresponding estimated consensus clustering  $\hat{\pi}_{cc}$ . The consensus clustering  $\hat{\pi}_{cc}$  is obtained by applying a clustering algorithm from the ensemble, e.g., k-means with the desired k to  $\mathbf{\hat{Z}}_{cr}$ . If the number of clusters is the same in all ensemble members  $(k_i = k_j, \forall e_i, e_j \in \mathcal{E})$ , we choose for  $\hat{\pi}_{cc}$  just the result of one of the ensemble members. The pseudocode of our algorithm is depicted in Algorithm 1. In Fig. 5, we have a visual illustration of one round of our DECCS algorithm applied to a synthetic data set, and Fig. 6 shows its optimization over several rounds.

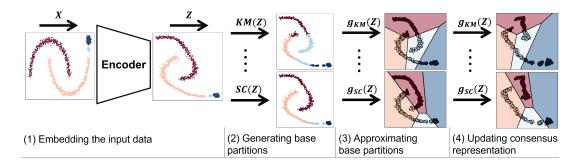


Fig. 5. Visualisation of one round of DECCS. (1) The encoder is used to embed data points X. (2) Clustering results are generated by applying ensemble members  $\mathcal{E} = \{\text{KM}, \dots, \text{SC}\}$  to Z. (3) Classifiers  $g_i$  are trained to predict the corresponding cluster labels  $\pi_i$  from Z. (4) Z is updated via minimizing  $\mathcal{L}$ .

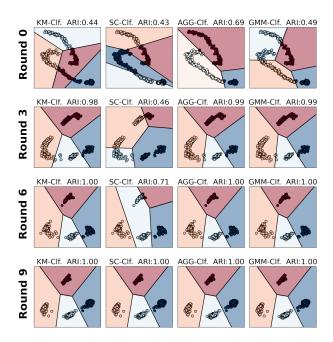


Fig. 6. Consensus representation learning with DECCS over several rounds for the synthetic data set and  $\mathcal{E} = \{KM, SC, AGG, GMM\}$ . Each plot shows the classification boundaries for each classifier (Clf.) trained on the cluster partitions. Over several rounds the clusters get better separated and more compact, leading to the same clustering for all ensemble members.

We use three heuristics for the optimization of DECCS. First, to speed up convergence we include the predicted cluster labels of the N - n unclustered data points for each classifier  $g_i$  during the computation of  $\mathcal{L}^i_{cons}$ . These predictions are updated during each mini-batch iteration for unclustered data points in the mini-batch  $\mathcal{B}$ . Second, to account for the classifiers' uncertainty we weight each distance computation in  $\mathcal{L}_{cons}^{i}$  with  $\alpha_{i,l} = g_{i,l}(enc_{\Theta}(\mathbf{x}))$ , which is the  $l^{th}$  entry of the prediction probability vector of classifier  $q_i$ . Third, to enforce the consensus over time t, we increase the weight of the consensus loss until a maximum weight  $\lambda_{cons}$  is reached. We use the sigmoid schedule as rampup function w, like [18], to increase the weight w(t) from 0 to  $\lambda_{cons}$ over time. In total, our algorithm needs the following userspecified parameters, an agreement threshold  $\tau$  that indicates how small the agreement gap between two subsequent sets of partitions should be. The data sampling size n, which should be chosen w.r.t. computational constraints and demands of clustering algorithms to have a sufficient number of samples. The maximum consensus weight  $\lambda_{cons}$  is a hyperparameter that together with the regularization weight  $\lambda_{rec}$  trades-off the confidence in the chosen ensemble with the structure of the underlying data. The maximum number of rounds T and the maximum number of mini-batch iterations ITER for the consensus representation update can be set based on computational constraints. We speed up the training of classifiers and encoders using early stopping, a heuristic that stops training once the loss on a held-out evaluation set starts to increase due to overfitting.

### V. RELATED WORK

## A. Consensus Clustering

Based on the consensus function (CF) consensus clustering methods can be broadly categorized into median partitionand object co-occurrence based methods. Median partition methods find a partition that is most similar to all the base partitions. Object co-occurrence based methods utilize the coassociation (CA) matrix to find the ideal partitioning, where the entries of this matrix reflect how often every two instances are partitioned together. Fred et al. [19] introduced Evidence Accumulation (EAC), a hierarchical clustering algorithm that uses entries of the CA matrix as a similarity measure that is used to produce the final clustering. More recently, [20] extended this idea by proposing Locally Weighted Evidence Accumulation (LWEA), introducing an entropy-based weighting schema, which makes it more robust to outlier partitions. Strehl et al. [9], and later Fern et al. [21], utilized the CA matrix to formulate graph-based algorithms as a consensus function. Li et al. [22] proposed a more efficient Nonnegative Matrix Factorization (NMF) based algorithm to factorize the CA matrix as an alternative.

To generate base partitions for high dimensional data, like images, a line of research follows the idea of random projections (RP). Inspired by the Johnson–Lindenstrauss (JS) lemma [23], [10] introduced with Random Projection Expectation Minimization (RP+EM) the first RP-based CC algorithm, where the data is projected onto various lower-dimensional subspaces using random matrices. The entries of the resulting CA matrix are then used for a hierarchical clustering approach. Similar to this idea, [11] proposed Random Projection Fuzzy c-Means (RP+FCM), where each subspace is clustered with a Fuzzy c-Means (FCM) algorithm. Those partitions are then combined with an agreement function. However, contrary to DECCS, RP methods are limited to linear transformations.

# B. Deep Clustering

Most, current DC methods are designed with only a single clustering model in mind, e.g., SpectralNet [7] for spectral clustering, DEC [1], IDEC [24], DCN [25] for k-means like clustering, VaDe [3] for Gaussian mixture models, or Deep-ECT [5] for hierarchical clustering or ENRC [26] for nonredundant clustering, see [27] for an overview. SpectralNet is a deep extension of spectral clustering for large data and out-ofsample generalization. DEC minimizes a soft auxiliary target distribution using the Kullback-Leibler divergence, which is related to soft k-means [28]. Improved DEC (IDEC) includes the AE reconstruction loss in the DEC objective to avoid arbitrary clustering results. In contrast to the soft clustering objective of DEC, the DCN algorithm uses hard cluster assignments together with an alternating optimization scheme. It alternates between k-means clustering and representation update to achieve a k-means friendly embedding. VaDE combines a Gaussian mixture model prior with a variational autoencoder (VAE) [29] to learn a deep generative clustering. DeepECT [5] introduced a deep embedded cluster tree to learn a hierarchical embedding.

The ConCURL [13] algorithm leverages image augmentation and RPs to learn a cluster ensemble of Softmax predictions to improve the overall clustering performance. A difference between their approach and ours is that they are limited to data that can be augmented, e.g., images or text. Further, they create a k-means like ensemble by using the Softmax, see [30] for the connection between the Softmax and k-means. In contrast to that, our DECCS algorithm can be used with a wide range of existing clustering methods and is not limited to k-means. Liu et al. [14] proposed the IEC algorithm which embeds multiple clustering results with a marginalized Denoising AE [31] and clusters the learned embedding with k-means, without considering the original data. Tao *et al.* [15] extended the idea of [14] and proposed the AGAE method. Instead of embedding the clustering results, AGAE uses a consensus graph constructed from the CA matrix of the base partitions as an input to a DC method, which together with the original data produces an enriched embedding. In contrast to our approach, AGAE does not learn a consensus with the neural network but uses initial clusterings to construct a consensus graph as input for their DC algorithm, without updating the graph during training. Importantly and in contrast to ConCURL [13] and DECCS, both IEC and AGAE are not jointly updating the consensus clusterings and representation, a key feature of DC [27] to improve cluster performance.

# VI. EXPERIMENTS

We evaluate our DECCS algorithm with respect to several aspects. In Section VI-A, we evaluate DECCS w.r.t. its most important hyperparameters for MNIST [32] as it is usually done in DC [1], [3], [24], [25] and show that our objective increases the agreement and cluster performance for all ensemble members across data sets. Additionally, we perform an ablation study across data sets. In Section VI-B, we compare DECCS to several CC and DC methods.

**Evaluation Metrics:** We evaluate the performance using normalized mutual information (NMI) [33] and adjusted rand index (ARI) [16]. Both range between 0 and 1, where 0 indicates no match and 1 a perfect match with the ground-truth. We evaluate the agreement within an ensemble by calculating the average pairwise NMI [9] between all clusterings.

**Data sets:** The synthetic data set (SYNTH) consists of four clusters and is depicted in Fig. 1a. The real-world data sets consist of commonly used DC image data sets like MNIST, Fashion-MNIST (FMNIST) [34], Kuzushiji-MNIST (KMNIST) [35], and USPS [36] and three UCI [37] data sets PENDIGITS, HAR and MICE. We provide a detailed description of the data sets in Appendix A. All data sets are preprocessed using a z-transformation.

**Experimental Setup:** For all data sets that have more than 2,000 data points, we use a feed-forward AE architecture with layers D-500-500-2000-10 for the encoder and a corresponding mirrored decoder, which is the same setting as used in [1]. For the MICE and SYNTH data sets, we have used smaller networks, with D-256-128-64 and D-20-20-2 for the encoders and mirrored decoders respectively. We use these architectures for DECCS, DEC, IDEC, DCN, and VaDE. For SpectralNet<sup>3</sup> and ConCURL<sup>4</sup>, we used the settings that are available in their public implementations. IEC and AGAE have no publicly available code, which is why we only show the NMI results reported in their papers<sup>5</sup>.

For hyperparameters that are specific to DECCS, we set  $\lambda_{cons}$  to 0.1 for the image data sets and to 10.0 for the UCI and SYNTH data sets, where the higher weight leads to better results for all data sets. The sampling size n is set to  $0.08 \cdot N$ for data sets with N > 11,000 and to  $0.5 \cdot N$  for the others. We let our algorithm run for T = 10 rounds and report the result with the highest agreement between ensemble members, thus not needing to specify  $\tau$ . We train the classifiers and encoder of DECCS with mini-batch SGD and momentum [38] set to 0.9 for all data sets and  $|\mathcal{B}| = 256$ . The classifiers are pretrained with a learning rate of 0.01 and the representation updates are done with a learning rate of 0.001, which is reduced by a factor of 0.9 after each round t. We set the number of maximum mini-batch iterations to ITER = 20,000for all data sets. We used the early stopping heuristic during the classifier pretraining and consensus representation learning

<sup>&</sup>lt;sup>3</sup>https://github.com/KlugerLab/SpectralNet

<sup>&</sup>lt;sup>4</sup>https://github.com/JayanthRR/ConCURL\_NCE

<sup>&</sup>lt;sup>5</sup>Symbol † indicates results are taken from [14] and ‡ from [15]

TABLE I Ablation study for combinations of loss terms of DECCS. Best results are marked as bold and runner-up is underlined. All results are given in NMI as mean  $\pm$  std over 10 runs.

$\mathcal{L}_{\text{cons}}$	L <sub>CE</sub>	$\mathcal{L}_{rec}$	SYNTH	MICE	PENDIGITS	HAR	MNIST	FMNIST	KMNIST	USPS
Х	X	x		$\begin{array}{c} 0.44 \pm 0.05 \\ 0.42 \pm 0.02 \\ 0.43 \pm 0.02 \end{array}$	$\begin{array}{c} 0.12 \pm 0.07 \\ 0.72 \pm 0.02 \\ 0.68 \pm 0.01 \end{array}$	$ \begin{vmatrix} 0.01 \pm 0.01 \\ 0.68 \pm 0.04 \\ 0.55 \pm 0.05 \end{vmatrix} $	$\begin{array}{c} 0.45 \pm 0.03 \\ 0.81 \pm 0.04 \\ 0.75 \pm 0.01 \end{array}$	$\begin{array}{c} 0.42 \pm 0.02 \\ 0.58 \pm 0.01 \\ 0.61 \pm 0.01 \end{array}$	$ \begin{vmatrix} 0.27 \pm 0.01 \\ 0.54 \pm 0.01 \\ 0.49 \pm 0.01 \end{vmatrix} $	
X X	X X	X X	$\begin{array}{c} 0.41 \pm 0.02 \\ \underline{0.82} \pm 0.09 \\ \textbf{0.99} \pm 0.01 \end{array}$	$\begin{array}{c} 0.50 \pm 0.04 \\ 0.43 \pm 0.04 \\ \underline{0.55} \pm 0.03 \end{array}$	$\frac{0.74}{0.73} \pm 0.02 \\ 0.73 \pm 0.02 \\ 0.82 \pm 0.02 $	$\begin{array}{c} 0.60 \pm 0.03 \\ 0.65 \pm 0.06 \\ \underline{0.73} \pm 0.03 \end{array}$	$\begin{array}{c} 0.79 \pm 0.01 \\ 0.83 \pm 0.02 \\ \underline{0.87} \pm 0.02 \end{array}$	$\begin{array}{c} \underline{0.64} \pm 0.01 \\ 0.63 \pm 0.02 \\ \underline{0.64} \pm 0.01 \end{array}$	$\begin{array}{c} 0.52 \pm 0.02 \\ 0.56 \pm 0.01 \\ \underline{0.60} \pm 0.01 \end{array}$	$\begin{array}{c} 0.71 \pm 0.01 \\ 0.79 \pm 0.01 \\ \underline{0.84} \pm 0.02 \end{array}$
Х	X	X	$0.99 \pm 0.02$	<b>0.57</b> ± 0.03	$\textbf{0.82}\pm0.02$	<b>0.75</b> ± 0.02	$\textbf{0.88} \pm 0.02$	$\textbf{0.65}\pm0.01$	<b>0.61</b> ± 0.01	<b>0.85</b> ± 0.01

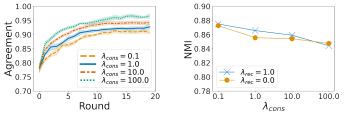


Fig. 7. DECCS parameter analysis for MNIST. (Left) Average agreement (thick lines) and 95% confidence intervals for ten runs of DECCS show that increasing the consensus weight  $\lambda_{cons}$  leads to an increased agreement between ensemble members during training. (Right) Average cluster performance for different values of  $\lambda_{cons}$  and  $\lambda_{rec}$  over ten runs.

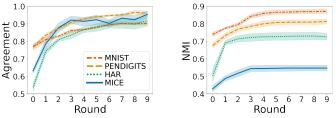


Fig. 8. Average agreement (thick lines) and 95% confidence intervals for ten runs of DECCS show the increase in agreement between ensemble members over training across data sets on the left side and the corresponding increase in cluster performance of DECCS on the right side.

and decreased the learning rate by 0.9 when a loss plateau was reached.

DEC, IDEC, and DCN are learning k-means friendly embeddings, SpectralNet extends spectral clustering, DeepECT learns hierarchical embeddings, and VaDE is a deep version of Gaussian mixture models. We, therefore, choose a heterogeneous ensemble of k-means (KM), spectral clustering (SC), agglomerative clustering (AGG), and Gaussian mixture models (GMM), based on the correspondence of the chosen DC methods, i.e.,  $\mathcal{E} = \{\text{KM}, \text{SC}, \text{AGG}, \text{GMM}\}$ .

For the CC approaches, we compare against eight methods (six classical methods, two utilizing RPs). We evaluate the CC methods on the raw and the AE embedded data sets using the same ensemble  $\mathcal{E}$  as DECCS. For all methods, we assume the number of clusters k to be known. We provide hyperparameter settings and further details for all methods in Appendix B. We uploaded the used data sets, our code and further results at https://gitlab.cs.univie.ac.at/lukas/deccs.

# A. Algorithm Evaluation

Ablation study: We evaluate the impact of the individual components of DECCS' loss function in Table I. We see that the combination of consensus loss ( $\mathcal{L}_{cons}$ ) and cross-entropy loss ( $\mathcal{L}_{CE}$ ), with and without reconstruction loss ( $\mathcal{L}_{rec}$ ) perform best (last two rows) for all data sets. Using only  $\mathcal{L}_{cons}$  without  $\mathcal{L}_{CE}$  leads to worse results because the classifiers are not preventing the merging of clusters (first row), as we have discussed in Fig. 4.

**Impact of ensemble size**: We evaluated the impact of the ensemble size by increasing its original size  $|\mathcal{E}| = 4$  by

doubling and tripling each member in the ensemble  $\mathcal{E}$ , leading to  $|\mathcal{E}_{\times 2}| = 8$  and  $|\mathcal{E}_{\times 3}| = 16$ . We evaluated the results by averaging the cluster performance of DECCS over ten runs on MNIST, where we achieved the same average NMI of 0.87 for each ensemble. We believe that we cannot see a benefit here because we use strong clustering methods, which do not add more diversity to the ensemble.

**Impact of sampling size** n: To evaluate the effect of the sampling size n, we varied it for ratios {0.02, 0.04, 0.06, 0.08} of MNIST (N = 70,000) and averaged the performance over ten runs. DECCS achieved the same performance (NMI = 0.87) for all ratios. We chose 0.08 for the remaining experiments as this was also stable for the other large data sets.

**Impact of**  $\lambda_{cons}$  and  $\lambda_{rec}$ : To demonstrate that the objective of DECCS increases the agreement between cluster ensemble members, we vary the consensus weight  $\lambda_{cons}$  for values in  $\{0.1, 1.0, 10.0, 100.0\}$  for the MNIST data set while keeping  $\lambda_{rec} = 0$ . We see on the left side of Fig. 7 that a higher value for  $\lambda_{cons}$  leads to a corresponding higher agreement. This is expected because we enforce the consensus with a higher weight. The right side of Fig. 7 shows the corresponding average cluster performance for  $\lambda_{rec} \in \{0.0, 1.0\}$ . The trend with and without the reconstruction loss is similarly downwards trending for very high values because a very highly weighted consensus loss disregards the underlying structure of the data. Increase of agreement and NMI during training: On the left side of Fig. 8, we show how DECCS increases the ensemble agreement over training for three UCI data sets respectively, and for MNIST as the behavior for the image data sets was very similar. This experiment gives additional evidence that our algorithm can effectively maximize the pairwise NMI

TABLE II
CLUSTER PERFORMANCE RESULTS MEASURED IN NMI. CORRESPONDING ARI VALUES ARE SHOWN IN TABLEIII.

Method	SYNTH	MICE	PENDIGITS	HAR	MNIST	FMNIST	KMNIST	USPS
DECCS	<b>0.99</b> ± 0.02	<b>0.57</b> ± 0.03	$ $ <b>0.82</b> $\pm$ 0.02	$0.75 \pm 0.02$	$\underline{0.88} \pm 0.02$	$\textbf{0.65}\pm0.01$	<b>0.61</b> ± 0.01	<b>0.85</b> ± 0.01
CSPA [9]	$0.75 \pm 0.00$	$0.35 \pm 0.02$	$0.73 \pm 0.02$	$0.49 \pm 0.02$	$0.59\pm0.04$	$0.53 \pm 0.03$	$0.46 \pm 0.02$	$0.65 \pm 0.02$
HGPA [9]	$0.75\pm0.00$	$0.37 \pm 0.01$	$0.61 \pm 0.05$	$0.49 \pm 0.04$	$0.47 \pm 0.02$	$0.46 \pm 0.02$	$0.35 \pm 0.02$	$0.56 \pm 0.03$
MCLA [9]	$0.57\pm0.08$	$0.34 \pm 0.01$	$0.73 \pm 0.04$	$0.59\pm0.03$	$0.55\pm0.05$	$0.53 \pm 0.02$	$0.49 \pm 0.02$	$0.60 \pm 0.08$
HBGF [21]	$0.62 \pm 0.02$	$0.33 \pm 0.03$	$0.72 \pm 0.03$	$0.48 \pm 0.03$	$0.58\pm0.03$	$0.51\pm0.04$	$0.46 \pm 0.02$	$0.64 \pm 0.01$
NMF [22]	$0.46 \pm 0.05$	$0.34 \pm 0.01$	$0.75\pm0.03$	$0.59\pm0.03$	$0.59\pm0.03$	$0.52\pm0.03$	$0.49 \pm 0.03$	$0.72 \pm 0.03$
LWEA [20]	$0.60 \pm 0.02$	$0.37 \pm 0.03$	$0.76 \pm 0.02$	$0.59\pm0.00$	$0.62\pm0.03$	$0.56\pm0.01$	$0.51 \pm 0.01$	$0.72 \pm 0.01$
RP+EM [10]	$0.61 \pm 0.00$	$0.46 \pm 0.04$	$0.67 \pm 0.05$	$0.46 \pm 0.06$	$0.48 \pm 0.05$	$0.50\pm0.05$	$0.44 \pm 0.04$	$0.64 \pm 0.04$
RP+FCM [11]	$0.64 \pm 0.05$	$0.28 \pm 0.08$	$0.63 \pm 0.01$	$0.51 \pm 0.01$	$0.22\pm0.02$	$0.40 \pm 0.01$	$0.25 \pm 0.01$	$0.38 \pm 0.02$
AE+CSPA [9]	$0.76 \pm 0.05$	$0.41 \pm 0.03$	$0.73 \pm 0.02$	$0.53 \pm 0.01$	$0.84 \pm 0.02$	$0.59 \pm 0.02$	$0.54 \pm 0.02$	$0.74 \pm 0.03$
AE+HGPA [9]	$0.75 \pm 0.00$	$0.41 \pm 0.03$	$0.64 \pm 0.07$	$0.49 \pm 0.05$	$0.61 \pm 0.02$	$0.50\pm0.03$	$0.43 \pm 0.03$	$0.60 \pm 0.02$
AE+MCLA [9]	$0.53 \pm 0.11$	$0.43 \pm 0.02$	$0.73 \pm 0.05$	$0.59\pm0.06$	$0.83 \pm 0.01$	$0.62\pm0.03$	$0.59 \pm 0.02$	$0.75 \pm 0.06$
AE+HBGF [21]	$0.65 \pm 0.08$	$0.40 \pm 0.03$	$0.71 \pm 0.04$	$0.53 \pm 0.03$	$0.83 \pm 0.02$	$0.58 \pm 0.02$	$0.54 \pm 0.02$	$0.72 \pm 0.02$
AE+NMF [22]	$0.50 \pm 0.13$	$0.44 \pm 0.04$	$0.76 \pm 0.04$	$0.60 \pm 0.01$	$0.82\pm0.04$	$0.61 \pm 0.02$	$0.59 \pm 0.03$	$0.82 \pm 0.04$
AE+LWEA [20]	$0.61 \pm 0.04$	$0.46 \pm 0.04$	$0.75\pm0.03$	$0.58\pm0.06$	$0.86 \pm 0.02$	$0.65 \pm 0.01$	<b>0.61</b> ± 0.03	$0.83 \pm 0.03$
AE+RP+EM [10]	$0.62\pm0.03$	$0.51 \pm 0.04$	$0.67 \pm 0.04$	$0.48 \pm 0.04$	$0.77\pm0.04$	$0.59\pm0.02$	$0.58 \pm 0.03$	$0.65 \pm 0.03$
AE+RP+FCM [11]	$0.63\pm0.10$	$0.41 \pm 0.03$	$0.54 \pm 0.05$	$0.48 \pm 0.03$	$0.45\pm0.07$	$0.49\pm0.03$	$0.31 \pm 0.04$	$0.37 \pm 0.03$
SpectralNet [7]	$0.72 \pm 0.06$	$0.27 \pm 0.06$	<b>0.82</b> ± 0.04	$0.61 \pm 0.06$	$0.92 \pm 0.00$	$0.64 \pm 0.01$	<b>0.61</b> ± 0.02	$0.83 \pm 0.02$
DEC [1]	$0.65 \pm 0.03$	$0.49 \pm 0.02$	$0.75 \pm 0.02$	$0.54 \pm 0.09$	$0.84 \pm 0.01$	$0.60 \pm 0.01$	$0.52 \pm 0.01$	$0.80 \pm 0.01$
IDEC [24]	$0.64 \pm 0.03$	$0.50 \pm 0.03$	$0.76 \pm 0.02$	$0.53\pm0.09$	$0.85\pm0.02$	$0.62\pm0.02$	$0.55 \pm 0.03$	$0.81 \pm 0.01$
DCN [25]	$0.59\pm0.08$	$0.48 \pm 0.04$	$0.75 \pm 0.02$	$0.51\pm0.08$	$0.84 \pm 0.03$	$0.62\pm0.02$	$0.54 \pm 0.04$	$0.78 \pm 0.04$
VaDE [29]	$0.62 \pm 0.10$	$0.45\pm0.06$	$0.75\pm0.02$	$0.54 \pm 0.09$	$0.83\pm0.03$	$0.65 \pm 0.01$	$0.56 \pm 0.01$	$0.79 \pm 0.03$
DeepECT [5]	$0.61 \pm 0.10$	$0.47 \pm 0.06$	$0.74 \pm 0.02$	$0.56 \pm 0.10$	$0.82\pm0.03$	$0.62\pm0.03$	$0.52 \pm 0.04$	$0.76 \pm 0.06$
ConCURL [13]	n.a.	n.a.	n.a.	n.a.	$0.60\pm0.04$	$0.48\pm0.02$	$0.30 \pm 0.03$	$0.49 \pm 0.02$
IEC [14]	-	-	$0.72^{\ddagger}$	-	$0.54^{\dagger}$	-	-	$0.64^{\dagger}$
AGAE [15]	-	-	$0.74^{\ddagger}$	-	-	-	-	$0.74^{\ddagger}$

between ensemble members (Eq. 2) by learning a consensus representation. The agreement is stabilizing for all data sets at round 8, except for MICE which fluctuates at a high agreement level due to the smaller data set size. The right side of Fig. 8 shows the corresponding increase in cluster performance. We see that DECCS reaches stable cluster performance already after round five for all data sets. ConCURL for all image data sets. ConCURL heavily relies on image augmentation and for small, greyscale images there are fewer augmentation invariances available, which might be the reason for ConCurl's poor performance. Further, ConCURL cannot be applied to non-augmentable data, which is why these results are marked as not applicable (n.a.).

### B. Cluster performance

In Table II, we show the clustering results of all methods w.r.t. NMI over ten runs. We see in Table II that for the SYNTH, MICE and HAR data set DECCS clearly outperforms the next best method. For the PENDIGITS data set, we perform similar to SpectralNet in NMI and outperform it w.r.t. ARI (0.73 vs 0.67). For the MICE data set, we see that all CC methods improve when applied to the AE embedded space, but DECCS is still outperforming them, showing that updating the representation can increase the cluster performance even further. The highest improvement for the real-world data sets can be seen for the HAR data set, where we outperform the next best clustering method (NMI=0.61) by 0.14. The results on the image data sets show that DECCS outperforms all comparison methods on USPS. DECCS performs similar to the DC methods for MNIST, FMNIST, and KMNIST. For MNIST, we are only outperformed by SpectralNet. Interestingly, the CC algorithms that are applied to the embedded space for the image data sets serve as strong baselines, e.g., reaching 0.65 and 0.61 NMI for FMNIST and KMNIST respectively. DECCS outperforms the deep consensus clustering method

### VII. DISCUSSION AND CONCLUSION

Noise and outlier points: Currently, we have not considered noise-aware clustering methods, like DBSCAN [39], in our ensembles. DECCS could be extended to include methods like DBSCAN, e.g., by excluding noise and outlier points during the representation update, such that a consensus representation is learned only for inlier clusters.

**Consensus representation learning**: With DECCS we have introduced the first algorithm to learn consensus representations for CC. In future work, we would like to explore alternative approaches for optimizing the proposed objective in Eq. 2, which could lead to novel approaches to CC.

We have proposed the idea of consensus representations, a novel way of learning a CC by maximizing the agreement between ensemble members using representation learning. Additionally, we have introduced the DECCS algorithm, to the best of our knowledge, it is the first DC algorithm that can use multiple heterogeneous clustering methods to jointly improve the learned representation and clustering results.

### REFERENCES

- 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.
- [2] S. P. Lloyd, "Least squares quantization in PCM," IEEE Trans. Inf. Theory, vol. 28, no. 2, pp. 129–136, 1982.
- [3] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," in *IJCAI*. ijcai.org, 2017, pp. 1965–1972.
- [4] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B*, vol. 39, no. 1, pp. 1–22, 1977.
- [5] D. Mautz, C. Plant, and C. Böhm, "Deep embedded cluster tree," in *ICDM*. IEEE, 2019, pp. 1258–1263.
- [6] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," WIREs Data Mining Knowl. Discov., vol. 2, no. 1, pp. 86–97, 2012.
- [7] U. Shaham, K. P. Stanton, H. Li, R. Basri, B. Nadler, and Y. Kluger, "Spectralnet: Spectral clustering using deep neural networks," in *ICLR* (*Poster*). OpenReview.net, 2018.
- [8] U. von Luxburg, "A tutorial on spectral clustering," *Stat. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.
- [9] A. Strehl and J. Ghosh, "Cluster Ensembles A Knowledge Reuse Framework for Combining Multiple Partitions," *Journal of Machine Learning Research*, pp. 583–617, 2002.
- [10] X. Z. Fern and C. E. Brodley, "Random projection for high dimensional data clustering: A cluster ensemble approach," in *ICML*. AAAI Press, 2003, pp. 186–193.
- [11] M. Popescu, J. Keller, J. Bezdek, and A. Zare, "Random projections fuzzy c-means (RPFCM) for big data clustering," in 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2015, pp. 1–6.
- [12] J. Bezdek, X. Ye, M. Popescu, J. Keller, and A. Zare, "Random projection below the JL limit," in 2016 International Joint Conference on Neural Networks (IJCNN). IEEE, 2016, pp. 2414–2423.
- [13] J. R. Regatti, A. A. Deshmukh, E. Manavoglu, and Ü. Dogan, "Consensus clustering with unsupervised representation learning," in *IJCNN*. IEEE, 2021, pp. 1–9.
- [14] H. Liu, M. Shao, S. Li, and Y. Fu, "Infinite ensemble for image clustering," in *KDD*. ACM, 2016, pp. 1745–1754.
- [15] Z. Tao, H. Liu, J. Li, Z. Wang, and Y. Fu, "Adversarial graph embedding for ensemble clustering," in *IJCAI*. ijcai.org, 2019, pp. 3562–3568.
- [16] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.
- [17] S. Monti, P. Tamayo, J. P. Mesirov, and T. R. Golub, "Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data," *Mach. Learn.*, vol. 52, no. 1-2, pp. 91–118, 2003.
- [18] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *NIPS*, 2017, pp. 1195–1204.
- [19] A. L. Fred and A. K. Jain, "Combining multiple clusterings using evidence accumulation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, 2005.
- [20] D. Huang, C.-D. Wang, and J.-H. Lai, "Locally Weighted Ensemble Clustering," *IEEE Transactions on Cybernetics*, pp. 1460–1473, 2018.
- [21] X. Z. Fern and C. E. Brodley, "Solving cluster ensemble problems by bipartite graph partitioning," in *Twenty-first international conference on Machine learning - ICML '04*. ACM Press, 2004, p. 36.
- [22] T. Li, C. Ding, and M. I. Jordan, "Solving Consensus and Semisupervised Clustering Problems Using Nonnegative Matrix Factorization," in *Seventh IEEE International Conference on Data Mining (ICDM* 2007). IEEE, 2007, pp. 577–582.
- [23] W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space," in *Contemporary Mathematics*, 1984.
- [24] 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.
- [25] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-meansfriendly spaces: Simultaneous deep learning and clustering," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 70. PMLR, 2017, pp. 3861–3870.

- [26] L. Miklautz, D. Mautz, M. C. Altinigneli, C. Böhm, and C. Plant, "Deep embedded non-redundant clustering," in AAAI. AAAI Press, 2020, pp. 5174–5181.
- [27] E. Aljalbout, V. Golkov, Y. Siddiqui, and D. Cremers, "Clustering with deep learning: Taxonomy and new methods," *CoRR*, vol. abs/1801.07648, 2018.
- [28] M. Jabi, M. Pedersoli, A. Mitiche, and I. B. Ayed, "Deep clustering: On the link between discriminative models and k-means," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 6, pp. 1887–1896, 2021.
- [29] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.
- [30] S. Hess, W. Duivesteijn, and D. Mocanu, "Softmax-based classification is k-means clustering: Formal proof, consequences for adversarial attacks, and improvement through centroid based tailoring," *CoRR*, vol. abs/2001.01987, 2020.
- [31] M. Chen, K. Q. Weinberger, F. Sha, and Y. Bengio, "Marginalized denoising auto-encoders for nonlinear representations," in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 32. JMLR.org, 2014, pp. 1476–1484.
- [32] 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.
- [33] X. V. Nguyen, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *J. Mach. Learn. Res.*, vol. 11, pp. 2837–2854, 2010.
- [34] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017.
- [35] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha, "Deep learning for classical japanese literature," *CoRR*, vol. abs/1812.01718, 2018.
- [36] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, 1994.
- [37] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml
- [38] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *ICML (3)*, ser. JMLR Workshop and Conference Proceedings, vol. 28. JMLR.org, 2013, pp. 1139–1147.
- [39] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*. AAAI Press, 1996, pp. 226–231.
- [40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html
- [42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR (Poster)*, 2015.

# Appendix

# A. DATA SETS

**Mice Protein Expression (MICE)** [37]: Data set consisting of 552 vectors with 77 dimensions and 8 ground-truth clusters. Each vector represents the expression levels of 77 proteins of the mice's cortex.

**Pendigits** [37]: Data set consisting of 10,992 vectors with 16 dimensions, representing 8 coordinates. The coordinates were gathered during the writing of digits (0 to 9) on a tablet.

**Human Activity Recognition (HAR)** [37]: Data set consisting of 10,299 vectors with 561 dimensions with records from smartphones and smartwatches. The data set contains six clusters corresponding to different human activities.

**MNIST** [32]: Data set consisting of 70,000 hand-written digits (0 to 9) with a size of  $28 \times 28$  pixels.

**FMNIST** [34]: Data set consisting of 70,000 goods from the Zalando online store. Each sample belongs to one of 10 products and has a size of  $28 \times 28$  pixels.

**KMNIST** [35]: Data set consisting of 70,000 Kanji characters (10 different characters) with a size of  $28 \times 28$  pixels.

**USPS** [36]: Data set consisting of 9,298 hand-written digits (0 to 9) with a size of  $16 \times 16$  pixels.

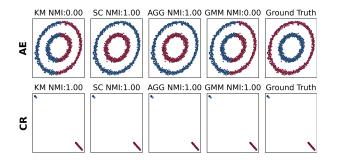


Fig. 9. Consensus representation (CR) learned with DECCS for two not linearly separable clusters. *Upper row*: Initial clustering results on an autoencoder (AE) embedding. Two ensemble members (SC, AGG) can perfectly discover the ground-truth clustering, but the two others (KM, GMM) do not perform better than chance. *Lower row*: Clustering results on a CR learned with DECCS leads to the same, perfect performance for all clustering algorithms

### B. EXPERIMENT SETUP

**Hardware setup:** We trained all DC algorithms on a machine with a single NVIDIA RTX 2080TI GPU (11GB onboard memory), 96 GB RAM, and an Intel(R) Xeon(R) Gold 6130 CPU. All other comparison methods were run on the same machine using only the CPU.

**Implementation:** We implemented DECCS in PyTorch (https: //pytorch.org/). Currently, the generation and approximation of base partitions is done sequentially, but could be further optimized using parallelization. One run of DECCS for MNIST took about one hour, which is in the same order of magnitude as other DC methods, e.g. DCN needed about 40 minutes.

**Parameters of cluster ensemble:** For the clustering algorithms in the ensemble  $\mathcal{E} = \{KM, SC, AGG, GMM\}$ , we used the implementations of the *sklearn* [40] package. For the real-world data sets, we parameterized them using this setting:

KMeans(n\_clusters=k)

For the synthetic data sets, we set linkage = 'single' for agglomerative clustering.

Consensus Clustering: For the classical methods, we used the same parameterization as suggested by the authors in the original papers ([9], [22], [21], [20]). For the RP-based methods, we determined the subspace dimension with a grid search and report the value with the best average NMI over 10 runs. We evaluated the RP algorithms with an ensemble size of four (size of our ensemble) and 30 (ensemble size used in [11]) and again picked the run with the highest NMI. The parameters of the Expectation Maximization (EM) and FCM algorithms were set according to [10] and [11] respectively. [11] proposed two versions of their RP-based algorithm, we run our experiments with both versions and only report the run with the highest NMI. Aligned with [11], we performed a grid search for q in [5, 100] for both RP-based methods and picked the run with the highest NMI. For LWEA, we set the hyperparameter  $\theta$  to 0.4, as [20] do for all their experiments. Except for LWEA, other classical methods are implemented using the cluster ensemble package.<sup>6</sup>

Deep Clustering: For the UCI data sets, we determined the learning rate and other AE parameters, like dropout rate [41], using a grid search during the AE pretraining taking the parameters with lowest reconstruction loss. All other AEs were pretrained with a learning rate of 0.001. All AEs were trained using early stopping and the learning rate was reduced by 0.5 if the reconstruction loss reached a plateau. We use for DECCS, VaDE, DEC, IDEC, and DCN a batch size of  $|\mathcal{B}| = 256$ . For VaDE, DEC, IDEC, and DCN, we use a constant learning rate of 0.0001 for the joint clustering. We train the DC algorithms for 100,000 mini-batch iterations for all data sets using the Adam optimizer [42] as was done in the original papers. For SpectralNet [7], we used for the SYNTH and MICE data set the same parameters as suggested for small data sets. For the other data sets (PENDIGITS, HAR, and image data), we used the same parameters as [7] used for MNIST.

**ConCURL** [13]: For ConCURL, we used the author's repository. For all data sets, we performed ten runs each. In each run, we trained the algorithm for 300 epochs. To change the basic architecture as little as possible, we transformed gray images into three dimensions by copying the gray color channel. In our experiment, we used PyTorch's resnet18 with a hidden MLP of 2048. We chose SGD as the optimizer with a

<sup>6</sup>https://github.com/827916600/ClusterEnsembles

### TABLE III

Cluster performance results measured in ARI. Best results are marked as bold and runner-up is underlined. All results are
Given in ARI as mean $\pm$ std over 10 runs. DECCS $\lambda_{\text{REC}} = 1$ indicates that $\mathcal{L}_{\text{REC}}$ was used, while $\lambda_{\text{REC}} = 0$ are the results without $\mathcal{L}_{\text{REC}}$ .

Method	SYNTH	MICE	PENDIGITS	HAR	MNIST	FMNIST	KMNIST	USPS
DECCS $(\lambda_{rec} = 1)$	$0.99 \pm 0.02$	<b>0.36</b> ± 0.04	$\textbf{0.73} \pm 0.03$	$\textbf{0.65}\pm0.03$	$\underline{0.85} \pm 0.05$	$0.47 \pm 0.01$	$\textbf{0.45}\pm0.01$	$\textbf{0.78} \pm 0.01$
DECCS $(\lambda_{rec} = 0)$	<b>1.00</b> ± 0.01	$0.33 \pm 0.03$	$0.72 \pm 0.04$	$0.62 \pm 0.03$	$0.85 \pm 0.04$	$0.47 \pm 0.01$	$0.44 \pm 0.02$	$0.77 \pm 0.02$
CSPA [9]	$0.67 \pm 0.00$	$0.19 \pm 0.02$	$0.64 \pm 0.04$	$0.37\pm0.02$	$0.49 \pm 0.05$	$0.40 \pm 0.04$	$0.36 \pm 0.02$	$0.54\pm0.02$
HGPA [9]	$0.67 \pm 0.00$	$0.20 \pm 0.01$	$0.47\pm0.08$	$0.36 \pm 0.05$	$0.34\pm0.03$	$0.30 \pm 0.03$	$0.23\pm0.03$	$0.42\pm0.04$
MCLA [9]	$0.43 \pm 0.08$	$0.17 \pm 0.01$	$0.61\pm0.06$	$0.46 \pm 0.03$	$0.42 \pm 0.04$	$0.36 \pm 0.02$	$0.35\pm0.02$	$0.52\pm0.08$
HBGF [21]	$0.56 \pm 0.02$	$0.17 \pm 0.03$	$0.64 \pm 0.03$	$0.37 \pm 0.04$	$0.49 \pm 0.03$	$0.39 \pm 0.05$	$0.37\pm0.03$	$0.54 \pm 0.02$
NMF [22]	$0.37 \pm 0.04$	$0.19 \pm 0.02$	$0.63 \pm 0.05$	$0.46 \pm 0.06$	$0.47\pm0.05$	$0.37 \pm 0.03$	$0.37\pm0.02$	$0.59\pm0.05$
LWEA [20]	$0.44 \pm 0.02$	$0.18 \pm 0.02$	$0.62\pm0.03$	$0.46 \pm 0.00$	$0.50\pm0.05$	$0.40 \pm 0.02$	$0.35\pm0.03$	$0.63 \pm 0.01$
RP+EM [10]	$0.44 \pm 0.00$	$0.26 \pm 0.04$	$0.42 \pm 0.11$	$0.31 \pm 0.07$	$0.22\pm0.04$	$0.29 \pm 0.07$	$0.24 \pm 0.05$	$0.45\pm0.07$
RP+FCM [11]	$0.53 \pm 0.10$	$0.15 \pm 0.05$	$0.47 \pm 0.02$	$0.32\pm0.00$	$0.12 \pm 0.02$	$0.25 \pm 0.01$	$0.14 \pm 0.01$	$0.20\pm0.06$
AE+CSPA [9]	$0.69 \pm 0.07$	$0.23 \pm 0.03$	$0.65\pm0.02$	$0.40 \pm 0.02$	$0.82 \pm 0.03$	$0.46 \pm 0.02$	$0.43 \pm 0.02$	$0.64 \pm 0.04$
AE+HGPA [9]	$0.67 \pm 0.01$	$0.23 \pm 0.02$	$0.51\pm0.08$	$0.36\pm0.05$	$0.47 \pm 0.04$	$0.34 \pm 0.04$	$0.29\pm0.03$	$0.45\pm0.03$
AE+MCLA [9]	$0.42 \pm 0.10$	$0.23 \pm 0.03$	$0.62\pm0.07$	$0.43 \pm 0.08$	$0.78\pm0.03$	$0.46 \pm 0.02$	$0.45 \pm 0.03$	$0.68\pm0.07$
AE+HBGF [21]	$0.59 \pm 0.10$	$0.22 \pm 0.03$	$0.62\pm0.05$	$0.41 \pm 0.03$	$0.80\pm0.03$	$0.44 \pm 0.02$	$0.44 \pm 0.02$	$0.64 \pm 0.03$
AE+NMF [22]	$0.41 \pm 0.15$	$0.25 \pm 0.04$	$0.66 \pm 0.07$	$0.41 \pm 0.03$	$0.75\pm0.08$	$0.48 \pm 0.03$	$0.44 \pm 0.05$	$0.74\pm0.09$
AE+LWEA [20]	$0.45 \pm 0.04$	$0.25 \pm 0.04$	$0.62 \pm 0.05$	$0.41 \pm 0.09$	$0.81 \pm 0.02$	$0.47 \pm 0.01$	$0.45 \pm 0.05$	$0.77 \pm 0.06$
AE+RP+EM [10]	$0.46 \pm 0.04$	$0.31 \pm 0.05$	$0.38\pm0.08$	$0.38\pm0.03$	$0.68\pm0.09$	$0.42 \pm 0.03$	$0.40 \pm 0.04$	$0.48 \pm 0.03$
AE+RP+FCM [11]	$0.54 \pm 0.16$	$0.22 \pm 0.03$	$0.37 \pm 0.04$	$0.35\pm0.04$	$0.29 \pm 0.08$	$0.30 \pm 0.03$	$0.18 \pm 0.04$	$0.21 \pm 0.05$
SpectralNet [7]	$0.53 \pm 0.08$	$0.15 \pm 0.04$	$0.67\pm0.08$	$0.46 \pm 0.09$	$0.93 \pm 0.00$	$0.47 \pm 0.00$	$0.42 \pm 0.04$	$0.67\pm0.05$
DEC [1]	$0.50 \pm 0.03$	$0.27 \pm 0.03$	$0.61 \pm 0.04$	$0.39 \pm 0.11$	$0.81 \pm 0.02$	$0.44 \pm 0.02$	$0.39\pm0.01$	$0.73 \pm 0.01$
IDEC [24]	$0.49 \pm 0.03$	$0.29 \pm 0.03$	$0.62\pm0.04$	$0.40 \pm 0.11$	$0.82\pm0.03$	$0.46 \pm 0.03$	$0.41 \pm 0.03$	$0.74\pm0.01$
DCN [25]	$0.44 \pm 0.08$	$0.27 \pm 0.04$	$0.60\pm0.04$	$0.36\pm0.11$	$0.79\pm0.06$	$0.45\pm0.03$	$0.38\pm0.05$	$0.70\pm0.07$
VaDE [29]	$0.49 \pm 0.13$	$0.25 \pm 0.05$	$0.61 \pm 0.04$	$0.38\pm0.10$	$0.78\pm0.06$	$0.48 \pm 0.02$	$0.40 \pm 0.02$	$0.70\pm0.06$
DeepECT [5]	$0.47 \pm 0.12$	$0.27\pm0.06$	$0.60\pm0.04$	$0.41\pm0.12$	$0.76\pm0.06$	$0.44 \pm 0.05$	$0.36\pm0.05$	$0.67\pm0.09$
ConCURL [13]	n.a.	n.a.	n.a.	n.a.	$0.48\pm0.05$	$0.34 \pm 0.02$	$0.20\pm0.03$	$0.33\pm0.02$

learning rate of 0.015 and set the batch size to 128. The alpha parameter was set to 0, and the beta and gamma parameters were set to 1. In the experiment, we set NCE-temp to 0.085 and NCE-k to 4096. For the hyperparameters, we followed the hyperparameters of CIFAR-10 available in their repository. For image augmentation, we used random rotations between -10 and 10 degrees, translations between 0 and 0.1, scaling between 0.6 and 1.2, and shearing between -10 and 10.