Deep Clustering With Consensus Representations

Lukas Miklautz1,2, Martin Teuffenbach1,2, Pascal Weber1,2, Rona Perjuci1,2, Walid Durani3, Christian Böhm1, Claudia Plant1,4
1Faculty of Computer Science, University of Vienna, Vienna, Austria
2UniVie Doctoral School Computer Science, Vienna, Austria
3Institute of Informatics, Ludwig Maximilian University of Munich, Munich, Germany
{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.1 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, and SpectralNet [6] for spectral clustering [7]. 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 [8]. Unfortunately, applying current CC methods to high-dimensional data sets leads to unsatisfactory results, because they are either limited to linear transformations [9]–[11], only work for $k$-means like clusterings [12], or only use CC information as input features for DC without updating the CC in response to improved data representations [13].

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
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 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) [14]. Applying existing DC methods, like DEC, VaDE, DeepECT or SpectralNet to the same synthetic data set using the same AE leads only to ARI values between 0.47 and 0.51. While this data set can be clustered by classical clustering techniques, DC methods fail, because their assumptions are not met.

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

In general, CC algorithms consist of two stages:

1) 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 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, [8] 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 \( \pi_{cc} = \arg\max_{\pi} \sum_{i=1}^{||\pi||} \text{NMI}(\pi_i, \pi) \). 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. Instead of the need to design a consensus function our DECCS algorithm learns a (non-linear) consensus function to learn the consensus representation.

III. OBJECTIVE FUNCTION FOR CONSENSUS REPRESENTATION LEARNING

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

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

\[
\min_{\Theta} \sum_{i=1,j>i}^{||\mathcal{E}||} \text{NMI}(e_i(\text{enc}_{\Theta}(X)), e_j(\text{enc}_{\Theta}(X))),
\]

with \( Z_{cr} := \text{enc}_{\Theta_{cr}}(X) \), where \( \text{enc}_{\Theta_{cr}} \) is the consensus representation function and \( c \) is a normalization constant \( c = \frac{2}{|E|^2-|E|} \) 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 [8]. The optimal encoder parameters for the consensus representation \( Z_{cr} \) are then learned with \( \Theta_{cr} = \arg\max_{\Theta} \min_{\mathcal{E}} \text{NMI}(e_i(\text{enc}_{\Theta}(X)), e_j(\text{enc}_{\Theta}(X))) \).

Note that Eq. 1 allows for degenerate solutions, like setting \( Z_{cr} \) to a constant if \( \text{enc}_{\Theta} \) is non-linear. To avoid degenerate solutions in practice we include regularizers in the objective, like enforcing the invertibility of \( Z_{cr} \) back to \( X \) 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. 1) of clustering results. DECCS works by alternating between representation and clustering update steps until an agreement is reached through the consensus representation.

We use a (non-linear) autoencoder (AE) to pretrain \( \text{enc} \) by reconstructing the original input data \( x \) from \( z \) using the decoder \( \text{dec} \) resulting in \( \hat{x} := \text{dec}(%5D(\text{enc}(x)) \). The AE reconstruction \( \hat{x} \) is learned by minimizing a reconstruction loss \( \mathcal{L}_{rec} = ||x - \hat{x}|| \), e.g., using the mean squared error. Given the pretrained AE, our DECCS algorithm consists of three main steps that we explain in the following sections.
A. Generating and approximating base partitions

At the beginning of each round \( t \) of our algorithm, we draw a small random sample \( X_t \) of size \( n < N \) from \( X \), because some clustering algorithms are impractical to be applied to large data sets and re-sampling can make the CC more robust [15]. Next, we embed the sample using \( \text{enc}_{\Theta}(X_t) = Z_t \) and generate a set of base partitions \( \Pi_t \) by applying all ensemble members to the embedding \( \pi_i = e_i(Z_t) \). 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 \( \Pi_t \).

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 \( G_i \), 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.,

\[
\mathcal{L}_{CE} = \sum_{i=1}^{||\Pi_t||} \mathcal{L}_{CE}^{i} = -\frac{1}{n} \sum_{j=1}^{n} \sum_{l=1}^{k_i} I[l = \pi_{i,j}] \log g_i(z_j),
\]

where \( \pi_{i,j} \) is the cluster label corresponding to the \( j \)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., \( g_i(x) = \text{softmax}(W_i x + b_i) \) with \( W_i \) and \( b_i \) 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.

B. Consensus Objective

Optimizing Eq. 1 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, we ”move” cluster points closer to their cluster representatives. Let \( C^j_{\pi} \) be the set of data points in the \( j \)th cluster of partition \( \pi_t \), then we can calculate its center \( \mu_j \) using \( \mu_j = \frac{1}{||C^j_{\pi_t}||} \sum_{x \in C^j_{\pi_t}} \text{enc}_{\Theta}(x) \), 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

\[
\mathcal{L}_{\text{cons}} = \sum_{i=1}^{||\Pi_t||} \mathcal{L}_{\text{cons}}^{i} = \sum_{i=1}^{||\Pi_t||} \| A_i M_i - \text{enc}_{\Theta}(X_t) \|_F^2
\]

\[
= \sum_{i=1}^{||\Pi_t||} \sum_{l=1}^{k_i} \left\| \mu_l - \text{enc}_{\Theta}(x) \right\|_2^2,
\]

where \( A_i \) is the \( n \times k_i \) one hot encoded cluster assignment matrix of partition \( \pi_i \). \( \| \cdot \|_2^2 \) the squared Euclidean norm, and \( \| \cdot \|_F \) the squared Frobenius norm. Here \( M_i \) and \( A_i \) are fixed, so the encoder \( \text{enc}_{\Theta} \) has to learn parameters \( \Theta \) that map embedded data points \( z = \text{enc}_{\Theta}(x) \) 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 cross-entropy loss (Eq. 2) in our objective, as we explain in the next Section.

C. Learning a consensus representation

Putting everything together the DECCS objective is

\[
L = \sum_{i}^{||\Pi_t||} \lambda_i (\mathcal{L}_{CE}^{i} + \mathcal{L}_{\text{cons}}^{i}) + \lambda_{\text{rec}} \mathcal{L}_{\text{rec}},
\]

where \( \lambda_i = \frac{1}{||\Pi_t||-1} \sum_{j=1}^{||\Pi_t||} \text{NMI}(\pi_t, \pi_j) \) is a weighting parameter based on the agreement of partition \( \pi_t \) 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 \( \mathcal{L}_{\text{rec}} \) as data dependent regularizer to avoid degenerate solutions by keeping \( Z \) approximately invertible and weight its importance with \( \lambda_{\text{rec}} \).

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. Eq. 4 can then be optimized using stochastic gradient descent. In the following section, we explain our algorithm in more detail.

D. Algorithm

Given a data set \( X \), the pretrained encoder \( \text{enc}_{\Theta} \), and a parameterized ensemble of clustering methods \( \mathcal{E} \), we learn a consensus representation \( Z_t \) and subsequently a consensus clustering \( \pi_{cc} \) with our DECCS algorithm in the following way. We encode the input data using \( \text{enc}_{\Theta} \), generate the base partitions by applying cluster ensemble members on \( Z_t \), approximate the base partitions using classifiers \( g_i \) and update the representation by minimizing \( L \). We repeat these steps 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 $\hat{Z}_{cr}$ 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 $\hat{Z}_{cr}$. We provide further illustrations and the pseudo code of DECCS in the full version.

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 $B$. Second, to account for the classifiers’ uncertainty we weight each distance computation in $\mathcal{L}^i_{cons}$ with $\alpha_{i,t} = g_{i,t}(\text{enc}_G(x))$, which is the $i^{th}$ entry of the prediction probability vector of classifier $g_i$. 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 [16], to increase the weight $w(t)$ from 0 to $\lambda_{cons}$ over time. In total, our algorithm needs the following user-specified 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

Fred et al. [17] introduced Evidence Accumulation (EAC), a hierarchical clustering algorithm that uses entries of the co-association (CA) matrix as a similarity measure that is used to produce the final clustering. More recently, [18] 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. [8], and later Fern et al. [19], utilized the CA matrix to formulate graph-based algorithms as a consensus function. Li et al. [20] 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 [21], [9] 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, [10] 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 [6] for spectral clustering, DEC [1], IDEC [22], DCN [23] for $k$-means like clustering, VaDE [3] for Gaussian mixture models, or DeepECT [5] for hierarchical clustering or ENRC [24] for non-redundant clustering. SpectralNet is a deep extension of spectral clustering for large data and out-of-sample generalization. DEC minimizes a soft auxiliary target distribution using the Kullback-Leibler divergence. 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. VaDE combines a Gaussian mixture model prior with a variational autoencoder (VAE) [25] to learn a deep generative clustering. DeepECT [5] introduced a deep embedded cluster tree to learn a hierarchical embedding.

The ConCURL [12] algorithm leverages image augmentation and RPs to learn a cluster ensemble of Softmax predictions to improve the overall clustering performance. Tao et al. [13] proposed the AGAE method. 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 [12] and DECCS, AGAE is not jointly updating the consensus clusterings and representation, a key feature of DC 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 as it is usually done in DC [1], [3], [22], [23] 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 evaluate DECCS to several CC and DC methods w.r.t. normalized mutual information (NMI) [26] and adjusted rand index (ARI) [14].

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), Kuzushiji-MNIST (KMNIST), and USPS and three UCI [27] data sets PENDIGITS, HAR and MICE. Experiment Setup: We use the same architectures for DECCS, DEC, IDEC, DCN, and VaDE. For SpectralNet and ConCURL, we used the settings that are available in their public implementations. AGAE has no publicly available code, which is why we only discuss the results reported in the paper. We provide the detailed setup in our full version.

Corresponding to the cluster models of DC methods DEC, IDEC, DCN, SpectralNet, DeepECT and VaDE, we choose an ensemble of $k$-means (KM), spectral clustering (SC), agglomerative clustering (AGG), and Gaussian mixture models (GMM), i.e., $\mathcal{E} = \{\text{KM}, \text{SC}, \text{AGG}, \text{GMM}\}$ as comparison.

For the CC approaches, we compare against eight methods (six classical methods, two utilizing RPs). We evaluate the CC methods on 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 CC results on the raw data, hyperparameter settings and further details for all methods in the full version of the paper. We uploaded the used data sets and our code 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 ($\mathcal{L}_{\text{cons}}$) and cross-entropy loss ($\mathcal{L}_{\text{CE}}$), with and without reconstruction loss ($\mathcal{L}_{\text{rec}}$) perform best (last two rows). Using only $\mathcal{L}_{\text{cons}}$ without $\mathcal{L}_{\text{CE}}$ leads to worse results because the classifiers are not preventing the merging of clusters (first row). The results for the remaining data sets are similar and can be found in the full paper.

Impact of hyperparameters: To demonstrate that the objective of DECCS increases the agreement between cluster ensemble members, we vary the consensus weight $\lambda_{\text{cons}}$ for values in $\{0.1, 1.0, 10.0, 100.0\}$ for the MNIST data set while keeping $\lambda_{\text{rec}} = 0$. We see on the left side of Fig. 3 that a higher value for $\lambda_{\text{cons}}$ leads to a corresponding higher agreement.

This is expected because we enforce the consensus with a higher weight. The right side of Fig. 3 shows the corresponding average cluster performance for $\lambda_{\text{rec}} \in \{0.0, 1.0\}$. The trend with and without the reconstruction loss is downwards trending, because a highly weighted consensus loss disregards the underlying structure of the data. Further, we found that the impact of the sampling size and ensemble size on DECCS for MNIST was small for reasonable ranges, more details are provided in the full paper version.

Increase of agreement and NMI: On the left side of Fig. 4, 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 between ensemble members (Eq. 1) by learning a consensus representation. The right side of Fig. 4 shows the corresponding increase in cluster performance. We see that DECCS reaches stable cluster performance already after round five for all data sets.

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). 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. DECCS performs similar to the DC methods for MNIST, FMNIST, and KMNIST and outperforms them on USPS. We clearly outperform the deep consensus clustering method AGAE for USPS (NMI=0.74) and PENDIGITS (NMI=0.74). For MNIST we are only outperformed by SpectralNet. Con-


| Method       | SYNTH | MICE | PENDIGITS | HAR | MNIST | FMINST | KMNIST | USPS |
|--------------|-------|------|-----------|-----|-------|--------|--------|------|
| DECCS [8]    | 0.99  | 0.57 | 0.82      | 0.75| 0.88  | 0.65   | 0.61   | 0.85 |
| AE+CSPA [8]  | 0.76  | 0.41 | 0.73      | 0.53| 0.84  | 0.59   | 0.54   | 0.74 |
| AE+HGPDA [8] | 0.75  | 0.41 | 0.64      | 0.49| 0.61  | 0.50   | 0.43   | 0.60 |
| AE+MCLA [8]  | 0.53  | 0.43 | 0.73      | 0.50| 0.59  | 0.62   | 0.59   | 0.75 |
| AE+HBBG [19] | 0.65  | 0.40 | 0.71      | 0.53| 0.83  | 0.58   | 0.54   | 0.72 |
| AE+NMF [20]  | 0.50  | 0.44 | 0.76      | 0.60| 0.82  | 0.61   | 0.59   | 0.82 |
| AE+LWEA [18] | 0.61  | 0.46 | 0.75      | 0.58| 0.86  | 0.65   | 0.61   | 0.83 |
| AE+RP+EM [9] | 0.62  | 0.54 | 0.67      | 0.48| 0.77  | 0.59   | 0.58   | 0.65 |
| AE+RP+FCM [10]| 0.65 | 0.41 | 0.54      | 0.48| 0.45  | 0.49   | 0.31   | 0.57 |

SpectralNet [6]  | 0.72  | 0.27 | 0.82      | 0.61| 0.92  | 0.64   | 0.61   | 0.82 |
| DEC [1]        | 0.65  | 0.49 | 0.75      | 0.54| 0.84  | 0.60   | 0.52   | 0.80 |
| DCN [23]       | 0.64  | 0.50 | 0.76      | 0.53| 0.74  | 0.62   | 0.50   | 0.82 |
| VaDE [25]      | 0.62  | 0.45 | 0.75      | 0.54| 0.84  | 0.63   | 0.59   | 0.79 |
| DeepECT [5]    | 0.61  | 0.47 | 0.74      | 0.56| 0.82  | 0.62   | 0.54   | 0.76 |
| ConCURL [12]   | n.a.  | n.a. | n.a.      | n.a| n.a.  | n.a    | n.a    | n.a  |

| Method       | SYNTH | MICE | PENDIGITS | HAR | MNIST | FMINST | KMNIST | USPS |
|--------------|-------|------|-----------|-----|-------|--------|--------|------|
| DECCS [8]    | 0.99  | 0.57 | 0.82      | 0.75| 0.88  | 0.65   | 0.61   | 0.85 |
| AE+CSPA [8]  | 0.76  | 0.41 | 0.73      | 0.53| 0.84  | 0.59   | 0.54   | 0.74 |
| AE+HGPDA [8] | 0.75  | 0.41 | 0.64      | 0.49| 0.61  | 0.50   | 0.43   | 0.60 |
| AE+MCLA [8]  | 0.53  | 0.43 | 0.73      | 0.50| 0.59  | 0.62   | 0.59   | 0.75 |
| AE+HBBG [19] | 0.65  | 0.40 | 0.71      | 0.53| 0.83  | 0.58   | 0.54   | 0.72 |
| AE+NMF [20]  | 0.50  | 0.44 | 0.76      | 0.60| 0.82  | 0.61   | 0.59   | 0.82 |
| AE+LWEA [18] | 0.61  | 0.46 | 0.75      | 0.58| 0.86  | 0.65   | 0.61   | 0.83 |
| AE+RP+EM [9] | 0.62  | 0.54 | 0.67      | 0.48| 0.77  | 0.59   | 0.58   | 0.65 |
| AE+RP+FCM [10]| 0.65 | 0.41 | 0.54      | 0.48| 0.45  | 0.49   | 0.31   | 0.57 |

### VII. CONCLUSION

We have proposed the idea of consensus representations, a novel way of learning a consensus clustering 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 deep clustering algorithm that can use multiple heterogeneous clustering methods to jointly improve the learned representation and clustering results.

### REFERENCES

[1] X. Jie, 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] 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.
[4] 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.
[5] T. Li, C. Ding, and M. J. Jordan, “Solving Consensus and Semi-supervised Clustering Problems Using Nonnegative Matrix Factorization,” in *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. IEEE, 2007, pp. 577–582.
[6] B. Johnson and J. Lindenstrauss, “Extensions of Lipschitz mappings into a Hilbert space,” in *Contemporary Mathematics, 1984*.
[7] Y. Guo, L. Gao, X. Liu, and J. Yin, “Improved deep embedded clustering with local structure preservation,” in *IJCAI*. ijcai.org, 2017, pp. 1753–1759.
[8] B. Yang, X. Fu, N. D. Sidiroopoulos, and M. Hong, “Towards k-mean-friendly spaces: Simultaneous deep learning and clustering,” in *ICML*, ser. Proceedings of Machine Learning Research, vol. 70. PMLR, 2017, pp. 3861–3870.
[9] M. Popescu, J. Keller, J. Bezdek, and A. Zare, “Random projections fuzzy c-means (RFFCM) for big data clustering,” in *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2015, pp. 1–6.
[10] 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.