Details (Don't) Matter: Isolating Cluster Information in Deep Embedded Spaces

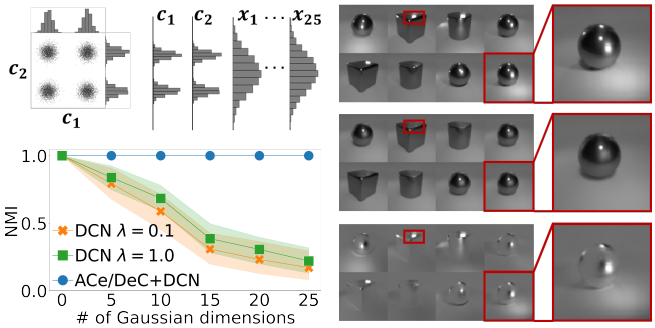
Lukas Miklautz[1,*], **Lena G. M. Bauer**[3,*], **Dominik Mautz**[2], **Sebastian Tschiatschek**[1,3], **Christian Böhm**[2,4] and **Claudia Plant**[1,3]
[1]Faculty of Computer Science, University of Vienna, Vienna, Austria
[2]Ludwig-Maximilians-Universität München, Munich, Germany
[3]ds:UniVie, Austria,
[4]MCML, Germany
{lukas.miklautz, lena.bauer}@univie.ac.at

Abstract

Deep clustering techniques combine representation learning with clustering objectives to improve their performance. Among existing deep clustering techniques, autoencoder-based methods are the most prevalent ones. While they achieve promising clustering results, they suffer from an inherent conflict between preserving details, as expressed by the reconstruction loss, and finding similar groups by ignoring details, as expressed by the clustering loss. This conflict leads to brittle training procedures, dependence on trade-off hyperparameters and less interpretable results. We propose our framework, ACe/DeC, that is compatible with **A**utoencoder **Ce**ntroid based **De**ep-**C**lustering methods and automatically learns a latent representation consisting of two separate spaces. The clustering space captures all cluster-specific information and the shared space explains general variation in the data. This separation resolves the above mentioned conflict and allows our method to learn both detailed reconstructions and cluster specific abstractions. We evaluate our framework with extensive experiments to show several benefits: (1) cluster performance – on various data sets we outperform relevant baselines; (2) no hyperparameter tuning – this improved performance is achieved without introducing new clustering-specific hyperparameters; (3) interpretability – isolating the cluster-specific information in a separate space is advantageous for data exploration and interpreting the clustering results; and (4) dimensionality of the embedded space – we automatically learn a low-dimensional space for clustering. Our ACe/DeC framework isolates cluster information, increases stability and interpretability, while improving cluster performance.

1 Introduction

The collection of massive amounts of complex data like images, text, video and audio gives rise to ever emerging challenges for data scientists trying to find patterns within data.



(a) (b)

Figure 1: **(a)** Impact of shared information on cluster performance. [top left] A synthetic 2D data set with four clusters, that can be perfectly clustered by all methods (step 0 on x-axis). [top right] Adding dimensions, which do not contain cluster specific information (e.g., unimodal Gaussian distributed), considerably hurts the performance of DCN (measured in average NMI and 95% confidence intervals over 20 runs). In contrast, DCN combined with our ACe/DeC framework remains stable as it can isolate the cluster information. **(b)** Our ACe/DeC framework applied to the OBJECTS data set, which consists of images of cubes, spheres, and cylinders under different lighting conditions. Our framework allows DCN to isolate the cluster-specific information (the shape), from shared information mainly relevant for reconstruction (the lighting). [top] Input sample. [middle] Cluster information: Reconstructions using only dimensions of the cluster space show distinct shapes, but have less light information. [bottom] Shared information: Reconstructions from the shared space dimensions contain only light information.

Often they work in an unsupervised setting without access to labels, as these might be very costly or impossible to obtain. We can learn these labels with clustering methods, which partition the data into similar groups. Unfortunately, clustering high-dimensional data such as images directly works unsatisfactorily. In such situations it is beneficial to combine clustering with deep learning [Xie *et al.*, 2016] to automatically learn a high-quality, low-dimensional representation for clustering. This quite recent field of research is termed *deep clustering* (DC). Autoencoder (AE)-based DC approaches, such as DEC [Xie *et al.*, 2016], IDEC [Guo *et al.*, 2017] or DCN [Yang *et al.*, 2017], in particular, are well-known and serve as building blocks for many other methods, see, e.g.,

[*]Authors with equal contribution

the survey by [Aljalbout *et al.*, 2018]. AEs are a common approach to learn non-linear embeddings of high-dimensional data. They consist of an encoder network, which maps the input $\mathbf{x}$ to a lower-dimensional space and a decoder network, which produces a reconstruction $\hat{\mathbf{x}}$ optimized for minimizing a reconstruction loss $\mathcal{L}_{\text{rec}} = ||\mathbf{x} - \hat{\mathbf{x}}||$. DC methods learn a 'cluster friendly' embedding by adding a clustering objective to the reconstruction loss. DEC, IDEC or DCN, e.g. are *centroid-based* as they use a k-means-like objective for 'moving' points closer to their respective centroids. They learn the reconstruction and clustering in a single embedded space, which blends the features necessary for clustering and reconstruction, leading to brittle performance and less interpretable results.

These drawbacks are due to an inherent conflict between the reconstruction loss of the AE, that tries to preserve all details and the clustering loss, that tries to abstract from details. One might say, "that an AE wants to learn everything a [clustering algorithm] wants to forget" [Epstein and Meir, 2019]. An instance of this conflict can be seen in the synthetic data set in Figure 1a. Adding features that are irrelevant for the clustering, but important for the reconstruction, hurts the clustering performance of DCN drastically. DCN and other DC algorithms that consider all features equally important for clustering and reconstruction, attempt to solve this issue by weighing the trade-off between cluster loss and reconstruction loss with a new hyperparameter λ. Importantly, automatically tuning λ is very hard if not impossible in the unsupervised setting and existing work leaves open on how to tune this parameter without access to ground truth labels. However, as we can see in Figure 1a DCN performs poorly, for low and high λ values. This shows that a hyperparameter based approach to address the balancing of reconstruction and clustering loss in general is not sufficient. Unfortunately, simply removing $\mathcal{L}_{\text{rec}}$ can lead to arbitrary clustering solutions [Guo *et al.*, 2017], which is why the reconstruction - clustering - dilemma needs to be approached differently.

In this work we propose to resolve the mentioned drawbacks with our novel ACe/DeC framework, that is compatible with **A**utoencoder **Ce**ntroid-based **De**ep **C**lustering methods. We rephrase the DC objective to account for both, cluster and non-cluster information. We do this by learning an individual weight for each dimension of the embedded space separating cluster-specific information from shared information, allowing us to learn good clusterings while keeping all the details. In Figure 1a we can see how our ACe/DeC framework removes the impact of irrelevant dimensions and improves the performance of DCN considerably. Another instance of irrelevant information for clustering is shown in Figure 1b. The OBJECTS data set consists of images of spheres, cubes and cylinders, with random lighting, where the light source is mostly relevant for reconstruction, but not for distinguishing between objects as it is shared across all images. Our ACe/DeC framework allows DCN to isolate the cluster-specific features from the ones that are shared as shown in the middle and bottom row of Figure 1b, respectively.

In the following, we will first introduce the general framework and how the architecture is designed to achieve the space separation. Then we apply ACe/DeC to one popular AE centroid-based DC procedure, namely DCN. We show experimentally the improvement of DCN with the framework compared to the original DCN algorithm as well as other competitors regarding

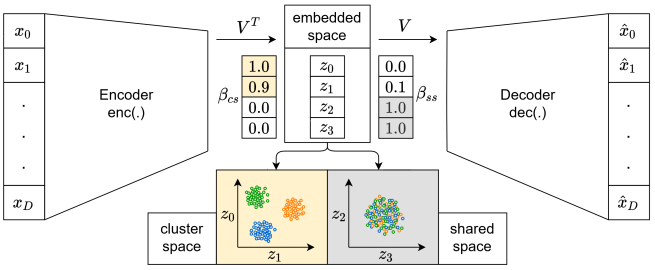


Figure 2: The overall architecture of ACe/DeC with the cluster space and the shared space. Our ACe/DeC framework extends the AE by introducing a learnable linear d $\times$ d transformation matrix $\mathbf{V}$ and two non-negative weight vectors β_{cs} and β_{ss} in the embedded space of the AE. The two weight vectors indicate which dimensions are important for clustering (β_{cs}) and which are not (β_{ss}). Here the β_{cs} weights indicate that the first two neurons z_0, z_1 have high cluster structure, while the last two neurons z_2, z_3 do not. In the plot below, one can see an example of a four dimensional embedded space split by ACe/DeC into two separate 2D spaces. We can see that the cluster space contains well-separated clusters, while the shared space contains only a single Gaussian without cluster structure. The matrix $\mathbf{V}$ aligns the clusters along the axes, based on the β-weights.

(1) cluster performance – on various data sets we outperform relevant baselines with increased stability regarding the choice of learning rate and cluster performance. Furthermore, we propose an optional augmentation procedure for image data, that can be used to further increase cluster performance and stability.

(2) hyperparameter tuning – most DC algorithms use cluster-specific hyperparameters that have to be tuned with ground truth labels. We remove such hyperparameters by rephrasing the DC objective to isolate cluster-specific information from shared information. This avoids having any data-specific hyperparameters to quantify the importance of cluster information making our approach more flexible to use in practice.

(3) interpretability – the isolation of cluster information leads to a natural separation of the embedded space into a cluster space and a shared space and enables a look into the AE blackbox. It is essential for visualizing and interpreting the clustering.

(4) dimensionality of the embedded space – ACe/DeC automatically determines the number of dimensions needed for clustering by removing irrelevant features.

2 Methodology

The AE is a key element of DC. It consists of an encoder network $\text{enc}(\cdot)$, which learns to project an input data point $\mathbf{x} \in \mathbb{R}^{\text{D}}$ from a data set X to an embedded (latent) vector $\mathbf{z} = \text{enc}(\mathbf{x})$ and a decoder network $\text{dec}(\cdot)$, which learns to project the embedded data point $\mathbf{z} \in \mathbb{R}^{\text{d}}$ back to the original

data space, resulting in the reconstruction $\hat{\mathbf{x}} := \mathrm{dec}(\mathbf{z})$ of $\mathbf{x}$. The dimension d of $\mathbf{z}$ is often chosen to be smaller than the input dimension D. This bottleneck architecture avoids that the AE simply learns to copy the input to the output, which would be possible for $\mathrm{d} \geq \mathrm{D}$, if no other regularization is added.

2.1 Method

To separate cluster information from non-cluster information we reformulate the commonly used DC objective $\mathcal{L} = \mathcal{L}_{\text{comp}} + \mathcal{L}_{\text{rec}}$. $\mathcal{L}_{\text{comp}}$ is the compression loss which is responsible for minimizing the distance of a point $\mathbf{z}$ to its closest centroid μ_k and $\mathcal{L}_{\text{rec}}$ is the AE reconstruction loss. Instead, we propose the loss function $\mathcal{L} = \mathcal{L}_{\text{cluster}} + \mathcal{L}_{\text{shared}} + \mathcal{L}_{\text{rec}}$, where the term $\mathcal{L}_{\text{shared}}$ is responsible for capturing the shared information by modelling the distance to the mean of the embedded data μ. In conjunction with the reconstruction loss this has the effect that we retain variance within the clusters, but are also able to model the common overall variance. $\mathcal{L}_{\text{cluster}}$ compresses the clusters similar to $\mathcal{L}_{\text{comp}}$. In particular we use $\mathcal{L} =$

$$\underbrace{\sum_{k=1}^{K}\sum_{\mathbf{z}\in C_k}\mathrm{dist}_{\beta_{cs}}(\mathbf{V^T z}, \mathbf{V^T}\mu_k)}_{\mathcal{L}_{\text{cluster}}} + \underbrace{\sum_{\mathbf{z}\in C}\mathrm{dist}_{\beta_{ss}}(\mathbf{V^T z}, \mathbf{V^T}\mu)}_{\mathcal{L}_{\text{shared}}} + \underbrace{\sum_{\mathbf{x}\in X}\mathrm{dist}_2(\mathbf{x}, \mathrm{dec}(\mathbf{V V^T z}))}_{\mathcal{L}_{\text{rec}}}, \quad (1)$$

where $\mathrm{dist}_{\beta_s}(\cdot)$, $s \in \{cs, ss\}$ are generic distance functions, e.g. for the squared euclidean distance ($\mathrm{dist}_2(\cdot)$) and for some d-dimensional vectors $\mathbf{h}, \mathbf{g}$ we define

$$\mathrm{dist}_{\beta_s}(\mathbf{h}, \mathbf{g}) = ||\mathbf{h} - \mathbf{g}||^2_{\beta_s} = \sum_{i=1}^{d} \beta_s[i]^2 (\mathbf{h}[i] - \mathbf{g}[i])^2. \quad (2)$$

Additionally, we have K clusters, with C_k as the set of all embedded data points in the k^{th} cluster with corresponding centroid μ_k. C is the set of all embedded data points in the single shared space cluster with centroid $\mu = \frac{1}{|C|}\sum_{\mathbf{z}\in C}\mathbf{z}$. We can now compute the assignment of objects to the given cluster centers in the cluster space by assigning them to their closest center $\arg\min_{k\in[1;K]} ||\mathbf{V}^T\mathbf{z} - \mathbf{V}^T\mu_k||^2_{\beta_{cs}}$.

We introduced new learnable parameters above. A linear $\mathrm{d} \times \mathrm{d}$ transformation matrix $\mathbf{V}$ and two non-negative weight vectors β_{cs} and β_{ss}. We explain the intuition behind them in Figure 2. To learn which dimensions are relevant for clustering, we need the β_s-weights to indicate which dimensions contain cluster structure, e.g. according to the k-means model, and which are not. Therefore, we use a trainable d-dimensional parameter $\mathbf{b}$ that is constrained by a sigmoid function $\beta_{cs}[i] = \mathrm{sigmoid}(\mathbf{b}[i]) := \frac{1}{1+\exp(-\mathbf{b}[i])}$ and $\beta_{ss}[i] = 1 - \beta_{cs}[i]$, where $[i]$ refers to the i^{th} component of a vector. Each component weighs one dimension of the embedded space. The two non-negative weight vectors β_{cs} and β_{ss} are constrained by $\beta_{cs} + \beta_{ss} = \vec{1}$, because the sigmoid function is between 0 and 1. These vectors represent a soft-assignment mechanism to the cluster space (cs) and shared space (ss), respectively.

The linear transformation matrix $\mathbf{V}$ can be seen as a linear layer which, 'guided' by the β-weights, axis-aligns the cluster structure along the most important cluster dimensions. With the above we can measure distances in two separated spaces using our weighted distances $\mathrm{dist}_{\beta_s}(\cdot)$, while still being differentiable. Altogether our ACe/DeC framework is lightweight requiring only $\mathrm{d}^2 + \mathrm{d}$ trainable parameters for the matrix $\mathbf{V}$ and weights β_s, where d is usually much lower than the original data dimensionality D. Before we apply ACe/DeC to DCN we first state the loss function of DCN $\mathcal{L} = \frac{\lambda}{2}\mathcal{L}_{\text{comp}} + \mathcal{L}_{\text{rec}}$

$$= \frac{\lambda}{2}\sum_{k=1}^{K}\sum_{\mathbf{z}\in C_k}\mathrm{dist}_2(\mathbf{z}, \mu_k) + \sum_{\mathbf{x}\in X}\mathrm{dist}_2(\mathbf{x}, \mathrm{dec}(\mathbf{z})) \quad (3)$$

$$= \frac{\lambda}{2}\sum_{k=1}^{K}\sum_{\mathbf{z}\in C_k}\|\mathbf{z} - \mu_k\|_2^2 + \sum_{\mathbf{x}\in X}\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2, \quad (4)$$

with the trade-off hyperparameter $\lambda > 0$. To integrate ACe/DeC into DCN, we adjust their loss function with our two new learnable parameters $\mathbf{V}$ and $\mathbf{b}$. We can then compute $\mathcal{L} = \mathcal{L}_{\text{cluster}} + \mathcal{L}_{\text{shared}} + \mathcal{L}_{\text{rec}}$ as

$$\sum_{k=1}^{K}\sum_{\mathbf{z}\in C_k}\|\mathbf{V^T z} - \mathbf{V^T}\mu_k\|^2_{\beta_{cs}} + \sum_{\mathbf{z}\in C}\|\mathbf{V^T z} - \mathbf{V^T}\mu\|^2_{\beta_{ss}} + \sum_{\mathbf{x}\in X}||\mathbf{x} - \mathrm{dec}(\mathbf{V V^T z})||_2^2, \quad (5)$$

where we can now learn which dimensions are important to the k-means clustering and which are not, without any potentially crucial hyperparameters. To further motivate this loss, one can think about the β_s-weighted euclidean distance from Eq.2 as rescaling the dimensions based on the cluster structure. If in the i^{th} dimension the k-means centroids represent the embedded data points better than a single centroid, $\beta_{cs}[i]$ will increase and $\beta_{ss}[i]$ will decrease accordingly in the next update step in order to lower the loss. We show this with a mathematical analysis by considering the gradient of the loss function with respect to $\mathbf{b}[i]$, which determines the update of the $\beta_s[i]$-weights. We present this and further analysis in the Supplement (SP) [Miklautz *et al.*, 2021] (Sec. 1.1 - 1.5).

Another motivation for combining ACe/DeC with DCN comes from SubKmeans [Mautz *et al.*, 2017]. We show in the SP (Sec. 1.6) that for a linear AE both methods optimize the same objective, because the reconstruction error will then force $\mathbf{V}$ to be orthogonal. With the ACe/DeC+DCN loss function we only update $\mathbf{V}$ and the space weights β_s; the update of centroids and cluster assignments for DCN is done separately. The original update procedure for DCN is alternating between the update of the AE parameters with fixed centroids and updating the cluster assignments and centroids with fixed AE parameters. We implemented another strategy, which allows for an update without alternation using mini-batch k-means [Sculley, 2010], for details see SP (Sec. 1.7).

2.2 Initialization and Augmentation Procedure

We initialize $\mathbf{V}$ as a random orthogonal matrix and randomly assign each dimension to one of the two spaces. We then perform k-means in the cluster space to get the initial centroids

μ_k. As the shared space is modeled with a single cluster, we initialize its centroid with the mean of the embedded data. This adds only little overhead and is similar to other k-means based DC algorithms. We then train $\mathbf{V}$ and $\mathbf{b}$ while holding everything else constant to get an initial estimate.

Optionally, our ACe/DeC framework allows to leverage domain knowledge in the form of data augmentation to increase clustering performance. For this we use $\mathbf{x}_{aug} = f(\mathbf{x})$, where the function f augments the input data, e.g., by rotating it. We then minimize the distance of the embedded augmented data point $\mathbf{z}_{aug} = \text{enc}(\mathbf{x}_{aug})$ to the centroid of the original data point μ_k, with $\text{dist}_{\beta_{cs}}(\mathbf{V^T}\mathbf{z}_{aug}, \mathbf{V^T}\mu_k)$. This forces the clusters to be invariant to the augmentation. Additionally, we make use of our cluster and shared space. We reconstruct the augmented data point $\mathbf{x}_{aug}$ from the shared space and the original $\mathbf{x}$ from the clustered space. With this we 'move' information unnecessary for clustering to the shared space.

3 Related Work

There are two clustering approaches, which are conceptually related to ACe/DeC through the idea of separating features containing clustering structure from features without clustering structure. The algorithms FOSSCLU [Goebl *et al.*, 2014] and SubKmeans [Mautz *et al.*, 2017] search for a single optimal subspace in the k-means sense, but are bound to linear relationships and do not scale well to high-dimensional data. In addition, their objectives are non-differentiable due to discrete subspace assignments making them unsuitable for DC with gradient-based optimization. Furthermore, they cannot be optimized jointly together with the AE, limiting their representational power and clustering performance.

AE-based DC methods are arguably the most prevalent ones [Aljalbout *et al.*, 2018; Min *et al.*, 2018]—whether the AE is only used for an initial representation of the data as in the DEC algorithm [Xie *et al.*, 2016], or the reconstruction and clustering objective is jointly optimized as in methods like IDEC [Guo *et al.*, 2017], DCN [Yang *et al.*, 2017], JULE [Yang *et al.*, 2016], DualAE [Yang *et al.*, 2019] or DEPICT [Dizaji *et al.*, 2017]. DEPICT is one of the few methods that does not introduce a cluster-specific hyperparameter. It uses a convolutional AE and leverages noise augmentation for a more robust clustering. IDEC combines a reconstruction loss with an auxiliary target distribution to minimize the Kullback-Leibler divergence such as in the DEC algorithm, weighing the trade-off with a hyperparameter. For DCN, network parameters, (hard) cluster assignments and centroids are updated in alternation. In each step of DCN the full data set is passed and a carefully weighted k-means loss is optimized together with the reconstruction loss. Recently, more powerful techniques that combine data-specific architectures with several cluster-specific hyperparameters have been introduced. These hyperparameters have to be tuned with ground truth labels, leaving open how to set them in an unsupervised setting. JULE combines a recurrent convolutional AE with a joint clustering objective, leveraging a well-tuned triplet loss. DualAE combines AE-based deep spectral clustering with mutual information maximization and noise augmentation.

A recently proposed probabilistic DC method is ClusterGAN [Mukherjee *et al.*, 2019], which combines a generative adversarial network (GAN) [Goodfellow *et al.*, 2014] with a clustering prior. While not an AE-based method, it uses similar to other existing methods the same space for capturing cluster and non-cluster information, leading to a trade-off between losses and less interpretable results. In contrast to our method, ClusterGAN can not distinguish features shared between all clusters from the ones that are cluster-specific, as can be seen in Figure 4 of [Mukherjee *et al.*, 2019]. All of these methods need three cluster-specific hyperparameters tuned by ground truth labels to balance the different loss terms, but they do not account for the situation that different dimensions might be less important to the clustering as we motivated in Figure 1a. This is in contrast to our approach, as we learn the importance of each dimension and assume that we have no access to ground truth labels, which is usually the case for clustering in practice. ACe/DeC shares its idea of soft-assigned feature spaces with [Ji *et al.*, 2017; Zhang *et al.*, 2018; Zhang *et al.*, 2019; Miklautz *et al.*, 2020]. However, in contrast to these (deep) subspace clustering methods, which find a separate subspace for each cluster or multiple clustering subspaces, we find a single cluster subspace for all clusters and a single non-cluster space for shared information. To our knowledge we are the first to introduce this idea to DC for resolving the clustering/reconstruction trade-off. Our framework is developed for centroid-based DC methods, which is why we do not compare to algorithms outside of this family. There are, of course, powerful classical clustering methods, e.g. HDBSCAN [Campello *et al.*, 2013]. However, the clustering notion is very different (density-based vs centroid-based).

4 Experiments

We evaluate all algorithms with six different data sets focusing on common DC benchmarks like MNIST [LeCun *et al.*, 1998], Fashion-MNIST [Xiao *et al.*, 2017] and USPS. Additionally, we use a data set based on real world images of traffic signs (GTSRB) [Houben *et al.*, 2013], recorded under different camera angles and daylight conditions, and two synthetic data sets to show the impact of irrelevant information on DC performance. The OBJECTS data set consists of $10,000$ synthetically generated gray scale images of spheres, cubes and cylinders with $4,096$ dimensions. The SYNTH-25 data set consists of $12,000$ data points with four well-separated Gaussian clusters in two dimensions and 25 independent, unimodal Gaussian distributed dimensions without cluster structure (27 dimensions in total), corresponding to step 25 of the x-axis in Figure 1a. Further explanations about the experiments, data sets, information on our hardware setup, compared methods, and all additional experiments are in the SP (Sec. 2). We uploaded our code and supplement at https://gitlab.cs.univie.ac.at/lukas/acedec_public.

4.1 Quantitative Experiments

In this section we show that by using our ACe/DeC framework, we can achieve stable cluster performance across data sets with varying degree of non-cluster information, without tuning of the hyperparameter λ and using less dimensions for clustering. For the image data sets we use for all methods 10 pretrained

Methods	SYNTH-25	FMNIST	MNIST-Full	MNIST-Test	OBJECTS	USPS	GTSRB
ACe/DeC+DCN	**1.00** ±0.00	**0.62** ±0.00	**0.89** ±0.01	**0.87** ±0.02	**1.00** ±0.00	**0.71** ±0.02	**0.52** ±0.04
DCN ($\lambda = 0.1$)	0.17 ±0.24	**0.62** ±0.02	0.87 ±0.01	0.86 ±0.02	**1.00** ±0.00	**0.71** ±0.03	0.19 ±0.05
DCN ($\lambda = 1.0$)	0.22 ±0.21	0.55 ±0.03	0.86 ±0.03	0.83 ±0.03	0.99 ±0.01	0.65 ±0.03	0.16 ±0.07
DCN ($\lambda = 10.0$)	0.29 ±0.22	0.55 ±0.04	0.70 ±0.08	0.66 ±0.05	**1.00** ±0.00	0.56 ±0.04	0.24 ±0.19
k-means	0.90 ±0.16	0.51 ±0.01	0.50 ±0.00	0.50 ±0.00	**1.00** ±0.00	0.61 ±0.00	0.22 ±0.01
SubKmeans	0.84 ±0.13	0.52 ±0.00*	0.48 ±0.01*	0.48 ±0.02*	-	0.61 ±0.00	-
AE + SubKmeans	0.11 ±0.10	0.59 ±0.02	0.80 ±0.01	0.79±0.01	0.98 ±0.02	0.65 ±0.02	0.47 ±0.02
AE + k-means	0.11 ±0.10	0.59 ±0.01	0.80 ±0.01	0.79±0.01	0.98 ±0.02	0.65 ±0.02	0.46 ±0.03

Methods + Aug	SYNTH-25	FMNIST	MNIST-Full	MNIST-Test	OBJECTS	USPS	GTSRB
ACe/DeC+DCN	N/A	**0.64** ±0.01	**0.94** ±0.00	0.94 ±0.00	**1.00** ±0.00	**0.86** ±0.02	**0.66** ±0.02
DCN ($\lambda = 0.1$)	N/A	0.60 ±0.02	**0.94** ±0.00	**0.95** ±0.01	**1.00** ±0.00	0.83 ±0.02	0.64 ±0.02
DCN ($\lambda = 1.0$)	N/A	0.57 ±0.03	**0.94** ±0.00	**0.95** ±0.01	**1.00** ±0.00	0.84 ±0.03	0.65 ±0.06
DCN ($\lambda = 10.0$)	N/A	0.57 ±0.03	0.92 ±0.02	0.94 ±0.02	**1.00** ±0.00	0.78 ±0.05	0.65 ±0.06

Table 1: NMI averages and standard deviations over 10 (20 for SYNTH-25) pretrained AEs. We compare the results with and without image augmentation in the upper and lower tables, respectively. The **upper** table shows that the performance of DCN depends in general quite heavily on λ. For SYNTH-25 and GTSRB the performance of DCN does not improve considerably for any of the considered λ values. This is mainly due to the AE, which focuses too much on reconstructing the irrelevant dimensions instead of preserving the cluster information as can be seen when comparing k-means and AE+k-means for GTSRB and SYNTH-25. This can also be seen for the OBJECTS data set, but the effect is much smaller. In contrast, ACe/DeC allows DCN to perform stable across all data sets. The **lower** table shows the performance of ACe/DeC and DCN leveraging image augmentation (random rotation and shifts). Unsurprisingly, both methods improve through domain knowledge in the form of augmentation, but DCN still depends on λ for FMNIST and USPS, where the learned augmentation invariances might not be as relevant. Importantly, image augmentation can not solve this problem for non-image data sets like SYNTH-25.

Methods	MNIST-Full	MNIST-Test	FMNIST	# HPs
ACe/DeC+DCN	0.94	0.94	0.64	**0**
DEPICT	0.92	0.92	0.39	**0**
DEC	0.74	0.75	0.57	**0**
IDEC	0.80	0.77	0.61	1
JULE	0.91	0.92	0.61	3
C-GAN	0.89	0.89	0.64	3
DualAE	0.94	0.95	0.65	3

Table 2: Average NMI comparison among recently proposed techniques on common benchmarks. DEC and IDEC have been rerun based on our re-implementations and other results taken from the literature. Even though the comparison methods have an unrealistic advantage—hyperparameters (HP) tuned with access to labels—our performance is equally strong.

Method	MNIST	FMNIST	USPS	GTSRB
SubKM	10*	10*	10	-
AE + SubKM	9	9	9	9
ACe/DeC+DCN	**6**	**5**	**5**	**4**

Table 3: Average number of dimensions found for all ten pretrained AEs with $d = 10$. The joint non-linear optimization allows our method to reduce the dimensionality by a factor of two compared to the linear method SubKmeans (SubKM).

fully connected AEs with an embedding size of $d = 10$ according to the procedure in [Xie *et al.*, 2016]. We use this basic architecture, because all methods would profit from a more powerful AE. We benchmark two implementations of our ACe/DeC framework with DCN. One which leverages augmentation invariances (random rotation and shifts) and one which does not. We compare this to DCN with different values of λ. We used the same settings for the training of all DC methods (training budget, learning rate, optimizer, etc., see the SP (Sec. 2.4) for details). Additionally, we compare to SubKmeans [Mautz *et al.*, 2017] (AE + SubKmeans), k-means (AE + k-means) and included the results of k-means and SubKmeans on the raw data sets as well. Results marked with * were run on a subset of $10,000$ objects, empty results were stopped due to run time constraints. Additionally, we compare our results against a reimplemented DEC/IDEC using the same pretrained AEs and recently introduced state of the art DC methods that leverage augmentation and data-specific architectures, namely DEPICT, JULE, ClusterGAN (C-GAN) and DualAE where we report the results from the respective papers. For SYNTH-25 and the experiments in Figure 1a, we trained for each setting 20 single layer linear AEs with $d = D$.

Stable cluster performance without hyperparameter tuning. Table 1 shows the Normalized Mutual Information (NMI) [Vinh *et al.*, 2010] results of the considered methods and data sets, where an NMI close to 1 indicates perfect clustering and 0 a random one. Our method performs stable across different data sets, while DCN alone fails for a data set like GTSRB, that contains several features unrelated to clustering. In our synthetic SYNTH-25 data set, where the number of irrelevant dimensions overtakes the number of dimension with cluster information, we can see that DCN's performance is considerably worse independent of its λ value. Augmentation improves the performance of our method and of DCN, but DCN still depends on λ for data sets where the learned invariances might not be as important, like FMNIST and USPS. We show that the cluster performance of DCN highly depends on the choice of λ and its learning rate in Figure 4, while ACe/DeC stabilizes DCN across different learning rates and data sets. In Table 2 we compare our method against recently reported results in the literature that use augmentation and data-specific neural network architectures. We outperform DEC, IDEC, DEPICT, JULE and ClusterGAN (C-GAN) and perform close to DualAE, despite being at a disadvantage as

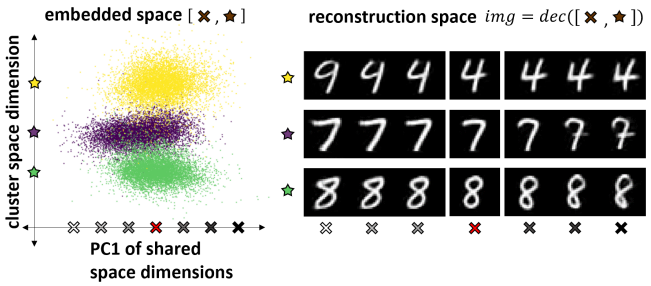
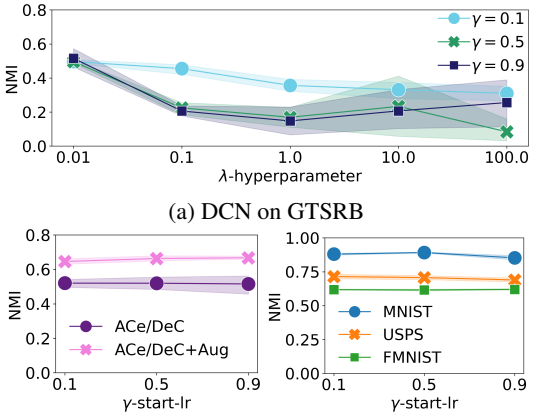


Figure 3: **Latent space traversal. (Left)** Plotting the dimension with most cluster structure (highest $\beta_{cs}[i]$) of ACe/DeC+DCN along the y-axis, we see that the digits 4,7 and 8 can be easily separated using a single dimension (colors indicate ground truth labels). The x-axis shows the direction of most variance in the shared space aka its first principal component (PC1). The equidistantly spaced markers ($\times$) along the PC1 axis indicate the traversal along this component. **(Right)** Each image shows the reconstructed data point with the corresponding coordinates in the embedded space plot on the left side, e.g., the top left image of the digit 4 is the reconstruction of the data point with the coordinates 'white cross' and 'yellow star'. Each row shows the traversal along PC1 for a fixed cluster space coordinate. Therefore, the cluster identity does not change, while the style varies as one moves from left to right, e.g., digit seven with and without horizontal bar. As style is a shared feature for digits, it is not contained in the cluster dimension, but can be captured in the shared space. These effects transfer to the other digits as well and other principal components reflect other features like rotation and thickness, showing that the shared space can capture multiple sources of variation at once, see SP (Sec. 2.1) for more details.



(a) DCN on GTSRB

(b) ACe/DeC on GTSRB (c) ACe/DeC on other DS

Figure 4: Figure 4a shows the average NMIs with 95% confidence intervals over 10 runs of DCN on GTSRB. We look at the stability of cluster performance w.r.t. different values of λ and different learning rates. We vary the learning rate with $10^{-3}\gamma$ for $\gamma \in \{0.1, 0.5, 0.9\}$, where the start value of 10^{-3} is based on the pretraining. We can observe, that the performance of DCN becomes very brittle. Its performance highly depends on different λ parameter and learning rate combinations. Furthermore, standard deviation is high for all parameter combinations. Using DCN with ACe/DeC does not need the λ hyperparameter, but we analyzed the performance for different learning rates in Figure 4b. The average NMIs are stable for all γ values with a standard deviation in a moderate range. The same holds for the other data sets (DS) as well, see Figure 4c.

we do not exploit ground truth labels.

Dimensionality of embedded space. Table 3 shows the average dimensions found with SubKmeans and ACe/DeC+DCN. The flexibility to learn a non-linear representation allows ACe/DeC to decrease the dimensionality needed for clustering by a factor of two. See SP (Sec. 1.4 and 2.2) on how to harden the soft assignments and more experiments.

4.2 Interpretability Experiments

In this section we show how we can use ACe/DeC to interpret which features are important for clustering and which are not. Using our framework, we gain three modes of understanding. First, we can separate the cluster and non-cluster information, as shown in Figure 1b. Second, we can traverse the latent space separately from the cluster space. With this we can view the change of varying information that is shared by all clusters. The analysis of the shared space of our algorithm when applied to MNIST can be seen in Figure 3. Here the direction of most variance of the shared space represents variation in style of MNIST digits. Note, that such an analysis is not possible for existing DC methods alone, which blend cluster and non-cluster information in a single space (see SP (Sec. 2.1) for examples). Third, by selecting the most discriminative dimensions based on β_{cs}, we can visualize the embedded space without the need of an additional dimensionality reduction technique like t-SNE [Maaten and Hinton, 2008], making our approach more faithful to the learned embedding as can be seen on the y-axis of Figure 3. We show another example for the OBJECTS data set in the SP (Sec. 2.1).

5 Conclusion

Sometimes details matter, sometimes they don't. Current AE-based DC methods need a priori knowledge about the data in the form of augmentation invariances or ground truth labels, to improve clustering results. However, this is an unrealistic setting for clustering and does not resolve the conflict between the autoencoder and clustering objective. We introduced our ACe/DeC framework that enables existing centroid-based DC algorithms to separate clustering information from shared information allowing the algorithm to preserve details for reconstruction and to abstract from details for clustering. Further, our framework improves interpretability, dimensionality reduction and performance stability, without a—in practice—impossible to tune hyperparameter. There are multiple promising directions to extend our framework, including but not limited to: integrating ACe/DeC into other centroid-based DC methods; replacing the reconstruction error with other self-supervised losses; or making our framework (semi-)supervised by combining it with a supervised AE [Le *et al.*, 2018].

Acknowledgments

This work was supported by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A and the Federal Ministry of Education, Science and Research of Austria (project: Digitize! Computational Social Sciences). The authors of this work take full responsibility for its content.

References

[Aljalbout *et al.*, 2018] Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, and Daniel Cremers. Clustering with deep learning: Taxonomy and new methods. *CoRR*, abs/1801.07648, 2018.

[Campello *et al.*, 2013] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[Dizaji *et al.*, 2017] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *ICCV*, pages 5747–5756, 2017.

[Epstein and Meir, 2019] Baruch Epstein and Ron Meir. Generalization bounds for unsupervised and semi-supervised learning with autoencoders. *CoRR*, abs/1902.01449, 2019.

[Goebl *et al.*, 2014] Sebastian Goebl, Xiao He, Claudia Plant, and Christian Böhm. Finding the optimal subspace for clustering. In *ICDM*, pages 130–139. IEEE Computer Society, 2014.

[Goodfellow *et al.*, 2014] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014.

[Guo *et al.*, 2017] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *IJCAI*, pages 1753–1759. ijcai.org, 2017.

[Houben *et al.*, 2013] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The german traffic sign detection benchmark. In *IJCNN*, pages 1–8. IEEE, 2013.

[Ji *et al.*, 2017] Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian D. Reid. Deep subspace clustering networks. In *NeurIPS,2017*, pages 23–32, 2017.

[Le *et al.*, 2018] Lei Le, Andrew Patterson, and Martha White. Supervised autoencoders: Improving generalization performance with unsupervised regularizers. In *NeurIPS*, pages 107–117, 2018.

[LeCun *et al.*, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Maaten and Hinton, 2008] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9:2579–2605, 2008.

[Mautz *et al.*, 2017] Dominik Mautz, Wei Ye, Claudia Plant, and Christian Böhm. Towards an optimal subspace for k-means. In *KDD*, pages 365–373. ACM, 2017.

[Miklautz *et al.*, 2020] Lukas Miklautz, Dominik Mautz, C Altinigneli, Christian Böhm, and Claudia Plant. Deep embedded non-redundant clustering. AAAI, 2020.

[Miklautz *et al.*, 2021] Lukas Miklautz, Lena Bauer, Dominik Mautz, Sebastian Tschiatschek, Christian Böhm, and Claudia Plant. Supplementary to: Details (don't) matter: Isolationg cluster information in deep embedded spaces. https://gitlab.cs.univie.ac.at/lukas/acedec_public, 2021.

[Min *et al.*, 2018] Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018.

[Mukherjee *et al.*, 2019] Sudipto Mukherjee, Himanshu Asnani, Eugene Lin, and Sreeram Kannan. Clustergan: Latent space clustering in generative adversarial networks. In *AAAI*, pages 4610–4617, 2019.

[Sculley, 2010] D. Sculley. Web-scale k-means clustering. In *WWW*, pages 1177–1178. ACM, 2010.

[Vinh *et al.*, 2010] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11:2837–2854, 10 2010.

[Xiao *et al.*, 2017] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[Xie *et al.*, 2016] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 478–487. JMLR.org, 2016.

[Yang *et al.*, 2016] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, pages 5147–5156. IEEE Computer Society, 2016.

[Yang *et al.*, 2017] Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *ICML*, volume 70, pages 3861–3870. PMLR, 2017.

[Yang *et al.*, 2019] Xu Yang, Cheng Deng, Feng Zheng, Junchi Yan, and Wei Liu. Deep spectral clustering using dual autoencoder network. In *CVPR*, pages 4066–4075. Computer Vision Foundation / IEEE, 2019.

[Zhang *et al.*, 2018] Tong Zhang, Pan Ji, Mehrtash Harandi, Richard I. Hartley, and Ian D. Reid. Scalable deep k-subspace clustering. In *ACCV (5)*, volume 11365 of *Lecture Notes in Computer Science*, pages 466–481. Springer, 2018.

[Zhang *et al.*, 2019] Tong Zhang, Pan Ji, Mehrtash Harandi, Wenbing Huang, and Hongdong Li. Neural collaborative subspace clustering. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, volume 97, pages 7384–7393. PMLR, 2019.