# DEEP CLUSTERING WITH ASSOCIATIVE MEMORIES

**Bishwajit Saha**
Department of CS
RPI
Troy, NY, USA
sahab@rpi.edu

**Dmitry Krotov**
MIT-IBM Watson AI Lab
IBM Research
Cambridge, MA, USA
krotov@ibm.com

**Mohammed J. Zaki**
Department of CS
RPI
Troy, NY, USA
zaki@cs.rpi.edu

**Parikshit Ram**
IBM Research
Yorktown Heights, NY, USA
Parikshit.Ram@ibm.com

## ABSTRACT

Deep clustering – joint representation learning and latent space clustering – is a well studied problem especially in computer vision and text processing under the deep learning framework. While the representation learning is generally differentiable, clustering is an inherently discrete optimization task, requiring various approximations and regularizations to fit in a standard differentiable pipeline. This leads to a somewhat disjointed representation learning and clustering. In this work, we propose a novel loss function utilizing energy-based dynamics via Associative Memories to formulate a new deep clustering method, **DCAM**, which ties together the representation learning and clustering aspects more intricately in a single objective. Our experiments showcase the advantage of **DCAM**, producing improved clustering quality for various architecture choices (convolutional, residual or fully-connected) and data modalities (images or text).

## 1 INTRODUCTION

The goal of clustering is to find coherent groups in a dataset. It is an important unsupervised learning task, and given the generality of the task, many different methods have been proposed for effective clustering (Xu & Tian, 2015; Zaki & Meira Jr, 2020). At a technical level, clustering critically relies on a notion of (pairwise) distance (or similarity) to distinguish pairs of data samples as being "similar" or "different", and the insights from clustering can be unintuitive or misleading without such a meaningful distance. When dealing with numerical data $S \subset \mathbb{R}^d$ with $d$ dimensions, metrics such as Euclidean distance are commonly used. Nevertheless, even with numerical data and an appropriate notion of distance, increasing data dimensionality (that is, increasing $d$) makes clustering computationally hard as well as conceptually difficult since the separation between similar pairs and dissimilar ones can start to vanish (Verleysen & François, 2005; Steinbach et al., 2004; Assent, 2012).

In various domains, both these problems manifest – first, the raw representation of samples can be extremely high dimensional (consider the number of pixels in an image, or the number of words in a vocabulary for a bag-of-words representation of documents); second, while we have an *ambient* representation, standard notions of vector distances (such as Euclidean) do not necessarily make sense – for example, Euclidean distance based on pixels can be large between an image and a slightly shifted version of it, which can be problematic if the content of an image is translation or rotation invariant.

One effective approach to handle these challenges is through *deep clustering* (Zhou et al., 2024), where the goal is to both learning a low dimensional *latent* space where standard distance metrics are meaningful, and to cluster or group the points at the same time. For the latent representations to be faithful to the original samples, deep clustering ensures that there is no significant information loss in the latent space, leading to the common use of autoencoders (AEs) (Rumelhart et al., 1985; Baldi, 2012; Bank et al., 2023) that learn latent representations (via an encoder) which can be used to reconstruct the original samples (via a decoder). The goal of deep clustering is to discover the cluster structure in the latent space while ensuring low reconstruction loss. This is a well

studied problem, especially in image datasets (Caron et al., 2018; Chang et al., 2017). While an autoencoder is usually differentiable, standard clustering schemes (such as $k$-means (MacQueen, 1967) or agglomerative (Johnson, 1967)) are inherently discrete methods, since *hard* clustering (where each sample is only assigned to a single cluster) is a discrete optimization problem. To incorporate it in a differentiable deep learning pipeline, clustering is often "softened" by allowing samples to be partially assigned to multiple clusters, although various "regularizations" push the soft assignments to match hard assignments approximately (Xie et al., 2016; Guo et al., 2017a). The recent `ClAM` (Saha et al., 2023) algorithm handles the dichotomy between hard assignments and differentiability via the use of associative memories, yielding an end-to-end differentiable clustering approach. Nevertheless, `ClAM` works only in the ambient $d$-dimensional data space, and is not designed to learn effective lower dimensional latent representations, which poses challenges when clustering high-dimensional data.

As noted above, deep clustering tackles the joint objective of learning a good latent representation where the points also cluster well. Whereas minimizing reconstruction loss is a prerequisite for deep representation learning, one option for clustering in latent space is to first *pretrain* an autoencoder to minimize the reconstruction loss, and then to freeze this latent space. Next, one can apply some clustering scheme to group the points in that (frozen) space. Many AE based existing deep learning methods adopt this scheme by either freezing both the encoder and decoder, or freezing only the decoder (Xie et al., 2016; Guo et al., 2017b; 2021; Chazan et al., 2019; Huang et al., 2023).

We propose a new approach called **`DCAM`** that uses Associative Memories (AM) as an inductive bias over the latent space that i) defines an (learnable) energy function over the latent space, and ii) utilizes the AM attractor dynamics to pull similar latent representations closer together. This leads to latent representations that are inherently well clustered without any explicit clustering objective, and the use of AM makes the whole process (i.e., learning the encoder, decoder, and cluster prototypes) end-to-end differentiable. Our key insight and contribution is that we seamlessly combine the clustering and reconstruction loss objectives into one expression that tackles the task of **clustering-guided latent representations**, whereas previous deep clustering methods considered these separately. Our work makes the following contributions:

- We propose **`DCAM`**, which uses associative memories to formulate a novel joint loss function that simultaneously learns effective representations and clusters in the latent space.
- We conduct a thorough evaluation on image and text datasets, demonstrating that **`DCAM`** significantly improves the clustering quality over both traditional (in ambient space) and deep clustering (in latent space) baselines.
- We show that **`DCAM`** retains superior representation quality as measured by the reconstruction loss; it is also agnostic to the encoder/decoder architecture choice.

## 2 Deep Clustering

Let $S \subset \mathbb{R}^d$ denote the input data in the ambient space, with an instance $x \in S$, and $[\![n]\!]$ a $n$-length index set $\{1, \ldots, n\}$. Deep clustering is an unsupervised task, where we have to learn (usually lower dimensional) representations such that (i) no (critical) information is lost in the latent lower dimensional representations, and (ii) the data in the latent space forms well-separated clusters. To ensure that no information is lost in the latent space, we learn an encoder $\mathbf{e} : \mathbb{R}^d \to \mathbb{R}^m$ ($m < d$) that maps the input $x \in \mathbb{R}^d$ to a latent space (that is, $\mathbf{e}(x) \in \mathbb{R}^m$), along with a decoder $\mathbf{d} : \mathbb{R}^m \to \mathbb{R}^d$ that maps the latent representation back to the original ambient space. Encoder $\mathbf{e}$ and decoder $\mathbf{d}$ together give us an autoencoder, and the loss of information is often measured as the *reconstruction loss*:

$$\mathcal{L}_r(\mathbf{e}, \mathbf{d}) = \sum_{x \in S} \ell_r(x, \mathbf{e}, \mathbf{d}) = \sum_{x \in S} \|x - \mathbf{d}(\mathbf{e}(x))\|^2$$

This loss term does not account for the cluster structure in the latent space. For that purpose, we consider $k$ cluster centers $\boldsymbol{\rho} = \{\rho_1, \ldots, \rho_k\} \subset \mathbb{R}^m$ in the latent space, so that the corresponding *clustering loss* is given by:

$$\mathcal{L}_c(\mathbf{e}, \boldsymbol{\rho}) = \sum_{x \in S} \ell_c(x, \mathbf{e}, \boldsymbol{\rho}) = \sum_{x \in S} \min_{i \in [\![k]\!]} \|\mathbf{e}(x) - \rho_i\|^2$$

which measures how close a sample is to its closest cluster center in the latent space with the $\min_{i\in[\![k]\!]}$ performed on a per-sample basis to denote the discrete assignment. A small value of $\mathcal{L}_c(\mathbf{e}, \boldsymbol{\rho})$ implies that all points in the latent space are close to their respective cluster centers.

Unsupervised deep clustering is often considered in the following form (Guo et al., 2017a;b; Cai et al., 2022)

$$\min_{\mathbf{e},\mathbf{d},\boldsymbol{\rho}} \mathcal{L}_r(\mathbf{e}, \mathbf{d}) + \gamma \mathcal{L}_c(\mathbf{e}, \boldsymbol{\rho}) \tag{1}$$

where $\gamma \geq 0$ is a hyperparameter that balances the clustering loss $\mathcal{L}_c$ and the reconstruction loss $\mathcal{L}_r$. This $\gamma$ plays a critical role in balancing the two terms in Eq. (1).

## 3 **DCAM**: Deep Clustering with AM Dynamics

Unlike existing deep clustering methods that minimize the objective in Eq. (1), which involves two separate components, namely the reconstruction loss in ambient space and clustering loss in latent space, in our novel **DCAM** formulation, we propose an AM-based approach that seamlessly combines these two aspects in a joint loss objective. Our approach not only updates the encoder and decoder models, but it also learns effective prototypes for clustering points in latent space.

### 3.1 Novel Loss Function

Fig. 1 shows the overall **DCAM** pipline. Given input data $S \subset \mathbb{R}^d$ in the ambient space, we first map input points $x \in S$ to the encoded point $v$ in the latent space, using the encoder model $\mathbf{e} : \mathbb{R}^d \to \mathbb{R}^m$ ($m < d$), that is, $v = \mathbf{e}(x)$. Next, assume that we are given $k$ memories or cluster prototypes (or centers) in latent space $\boldsymbol{\rho} = \{\rho_1, \rho_2, \ldots, \rho_k\}$, with $\rho_i \in \mathbb{R}^d$, we employ AM dynamics to update the representation of latent points $v$, using $T$ recursive steps, which we denote via the attractor dynamics operator $A^T_{\boldsymbol{\rho}}(v)$, so that $v' = A^T_{\boldsymbol{\rho}}(v)$. We discuss the details of the AM dynamics operator in Section 3.2 below, but it essentially tries to move the encoded point $v$ closer to a prototype $\rho_i$. Finally, given the updated latent representation $v'$, we employ the decoder model $\mathbf{d} : \mathbb{R}^m \to \mathbb{R}^d$ to map it back to the original ambient space, i.e., $x' = \mathbf{d}(v')$.

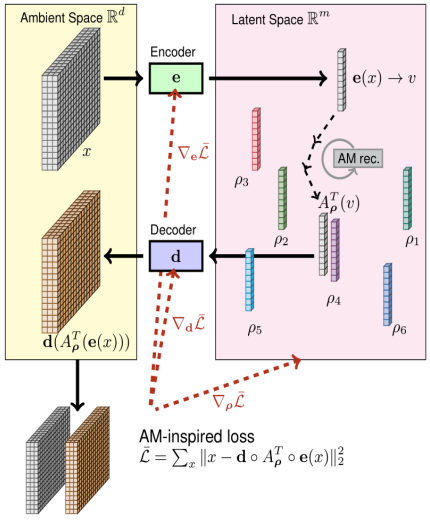



Figure 1: **DCAM**: AM-enabled deep clustering. The solid arrows $\longrightarrow$ denote the forward-pass to compute the single loss term in Eq. (2). The dashed arrows $- \rightarrow$ denote the backward pass showing the single loss driving all updates.

A key feature of **DCAM** is that the whole pipeline is differentiable, and all the components, namely the encoder parameters, the latent cluster prototypes $\boldsymbol{\rho}$, and the decoder parameters are learnable. Our novel joint loss function that combines both the clustering and reconstruction aspects into a single expression is defined as:

$$\min_{\mathbf{e},\mathbf{d},\boldsymbol{\rho}} \bar{\mathcal{L}}(\mathbf{e}, \mathbf{d}, \boldsymbol{\rho}) = \sum_{x \in S} \left\| x - \mathbf{d}\left(A^T_{\boldsymbol{\rho}}\left(\mathbf{e}\left(x\right)\right)\right)\right\|^2 \tag{2}$$

Here AM becomes the intricate part of the encoder that transforms the embedding space (obtained by the encoder) into a clustering-friendly new space to find clusters (as opposed to the existing deep clustering schemes that use different additional loss functions, e.g., clustering loss in Eq. (1) and/or regularizations to get a similar effect). This AM enabled *novel deep clustering loss* $\bar{\mathcal{L}}$ is a single term that elegantly combines all the parameters in the deep learning pipeline – for the encoder $\mathbf{e}$, the cluster centers $\boldsymbol{\rho}$ and the decoder $\mathbf{d}$.

### 3.2 Energy Dynamics in Latent Space

We now give details of how the attractor dynamics in latent space. Given the $k$ cluster prototypes $\boldsymbol{\rho} = \{\rho_1, \ldots, \rho_k\}, \rho_i \in \mathbb{R}^d$, and an encoded latent point $v \in \mathbb{R}^d$, the energy function for $v$ is defined as (Saha et al, 2023):

$$E(v) = -\frac{1}{2\beta} \log\left(\sum\nolimits_{i\in[\![k]\!]} \exp(-\beta \left\|\rho_i - v\right\|^2)\right)$$

where the scalar $\beta > 0$ denotes inverse temperature, so that as $\beta$ increases the $\exp(\cdot)$ function emphasizes the leading term, suppressing the others. *The attractor dynamics are then driven by gradient descent on the energy landscape*. That this, with $v^0 = v$, the updated representation of $v^t$ is given as

$$v^{t+1} = A_{\boldsymbol{\rho}}(v^t) = v^t - \tau \, \nabla_v E$$

where $\tau > 0$ is the step size that determines how quickly the latent point moves on the energy landscape, and

$$\nabla_v E = \frac{\partial E(v)}{\partial v} = \sum_{i \in [\![k]\!]} (\rho_i - v) \, \mathsf{softmax}(-\beta \|\rho_i - v\|^2)$$

with $\mathsf{softmax}(-\beta\|\rho_i - v\|^2) = \frac{\exp(-\beta\|\rho_i - v\|_2^2)}{\sum_{j \in [k]} \exp(-\beta\|\rho_j - v\|_2^2)}$.

Thus, the AM operator $A_{\boldsymbol{\rho}}(v^t)$ denotes the new latent vector obtained by updating $v^t$. Further, we use the notation $A_{\boldsymbol{\rho}}^T(v)$ to denote the dynamics for $T$ steps, i.e., $v' = A_{\boldsymbol{\rho}}^T(v) = A_{\boldsymbol{\rho}}(A_{\boldsymbol{\rho}}(\cdots A_{\boldsymbol{\rho}}(v)))$, where the operator $A_{\boldsymbol{\rho}}$ is applied to $v$ recursively for $T$ steps to obtain the updated representation $v'$. The attractor dynamics ensure that every memory or prototype $\rho_i, i \in [\![k]\!]$, forms a "basin of attraction", and with enough recursions $T$, any latent point $v$ will usually converge to exactly one of these memories $\rho_i$, which thus act as cluster centers. Further, the recursive dynamics is differentiable, with the memories learned via standard backpropagation.

## 4 Empirical Evaluation

We evaluate the performance of **DCAM** on a diverse set of 8 datasets (6 images and 2 text sets), ranging in size from 296 to 49152 (raw) features and containing 2007 to 60000 samples. The selection of the number of clusters for each dataset is based on its intrinsic class count, with no reliance on class information during clustering or hyperparameter selection (see dataset details in Appendix B.2). We conduct a comparative analysis of **DCAM** against established clustering methods, including $k$-means (Lloyd, 1982), agglomerative clustering (or Agglo.) (Müllner, 2011), `ClAM` (Saha et al., 2023), DCEC (Guo et al., 2017b), DEKM (Guo et al., 2021) and EDCWRN (or EDC) Oskouei et al. (2023), SCAN (Van Gansbeke et al., 2020) and NNM (Dang et al., 2021). Detailed parameter setting of the networks are in Appendix B.4, while implementation details are in Appendix B.5.

Table 1: Per-method best SC across all architectures (while RRL is within 10% of the respective pretrained AE loss), comparing **DCAM** to baselines. Best for each dataset is in bold. *Higher SC is better, but lower RRL is better.* The top set of rows are vision datasets, and the bottom set are text datasets. A '-' indicates not applicable (NA); e.g., DCEC, DEKM, SCAN, NNM work only on image datasets. Further, we report SCAN and NNM results only on C-10, C-100 and STL, since these are the only datasets for which pretrained contrastive encoders are available. x$^{\blacktriangledown}$ indicates negative RRL which means the RL of the method is x% less than the pretrained AE loss.

| Dataset | SC | | | | | | | | | RRL | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $k$-means | Agglo. | `ClAM` | DCEC | DEKM | EDC | SCAN | NNM | **DCAM** | DCEC | DEKM | EDC | **DCAM** |
| FM | 0.257 | 0.201 | 0.279 | 0.923 | 0.260 | 0.483 | - | - | **0.970** | 9.8 | **13.9**$^{\blacktriangledown}$ | 10 | 1.6$^{\blacktriangledown}$ |
| C-10 | 0.084 | 0.372 | 0.208 | 0.787 | 0.116 | 0.511 | 0.541 | 0.587 | **0.863** | 9.6 | 8.6 | 10 | **19.5**$^{\blacktriangledown}$ |
| C-100 | 0.015 | 0.149 | 0.053 | 0.470 | -0.007 | 0.311 | 0.321 | 0.358 | **0.598** | 7.5 | **34.3**$^{\blacktriangledown}$ | 10 | 1.4$^{\blacktriangledown}$ |
| USPS | 0.195 | 0.158 | 0.194 | **0.935** | 0.217 | 0.461 | - | - | 0.891 | **5.3**$^{\blacktriangledown}$ | 4.3 | 0.0 | 8.7 |
| STL | 0.079 | 0.270 | 0.108 | 0.259 | 0.082 | 0.411 | 0.552 | 0.540 | **0.891** | 9.2 | 0.6 | **4.9**$^{\blacktriangledown}$ | 10 |
| CBird | -0.019 | 0.094 | -0.026 | 0.311 | -0.032 | 0.171 | - | - | **0.448** | 10 | **0.0** | 10 | 9.1 |
| R-10k | -0.010 | 0.114 | -0.002 | - | - | 0.023 | - | - | **0.564** | - | - | 10 | **10** |
| 20NG | -0.021 | 0.114 | -0.008 | - | - | 0.101 | - | - | **0.197** | - | - | 10 | **10** |

**Comparison with Baselines:** We present the best Silhouette Coefficient or SC achieved (while constraining the reconstruction loss or RL to be within 10% of the pretrained AE loss) for **DCAM** and the baselines for all 8 datasets in Table 1. As it is hard to compare the raw RL numbers if the base AE is different for different methods, we consider relative RL (RRL) defined as $(RL - RL\_PAE)/RL\_PAE)$ where $RL\_PAE$ is the pretrained/base RL. We report the best SC per method with $RRL <= 10\%$.

From Table 1, we see across both image and text datasets, **DCAM** consistently outperforms traditional and deep clustering baselines in terms of SC while keeping RRL relatively low. To provide a

comprehensive view alongside SC, we also present the best RRL results (while constraining the SC to be within 10% of the best/peak SC of the method) in Table 3 in Appendix. Note that SCAN and NNM do not have a reconstruction loss term as they work on the pretrained (pretext) model by SimCLR (Chen et al., 2020) and utilize only the encoder (discarding the decoder) for clustering purpose. We observe that **DCAM** has the best SC values across all datasets except for USPS where DCEC performs the best. Its RL remains competitive. These results demonstrate that **DCAM** excels not only in achieving the best SC but also in simultaneously minimizing RL compared to the baselines. It is important to note that the `ClAM`, $k$-means and Agglomerative clustering results are the best from either using no autoencoder, or those from applying them in latent space on the points obtained from the pretrained autoencoder. In particular, we can see that **DCAM** vastly outperforms a straightforward application of `ClAM` in latent space.

For additional insights, in Appendix C.2, we present the best SC (while keeping RL within 10% of the pretrained AE loss) and its corresponding NMI, RL, cluster size and balance metrics obtained by all schemes in Table 9, and in Table 10 we report the best RL (while keeping SC within 10% of the best SC of the method) and its associated SC, NMI, and other metrics. Finally, in Table 11 we further report the best NMI obtained along with the associated SC, RL, and other metrics. These results clearly show that **DCAM** offers the best clustering performance in terms of SC, as well as having low reconstruction loss. It also performs very well on the supervised NMI metric. In fact, for NMI, **DCAM** has the best value in 5 out of the 8 datasets (see Table 11).

**Effect of AE Architecture:** Table 4 (in Appendix) shows that the performance improvement achieved by **DCAM** is independent of the Autoencoder (AE) architecture choice. **DCAM** with *all three architectures* – CAE, EAE, and RAE – consistently outperforms their respective baselines, DCEC, DEKM and EDCWRN with similar architecture. That is, within each type of AE, **DCAM** has better results than DCEC and DEKM, or EDC. This not only underscores the superiority of the internal algorithm of **DCAM** over the corresponding baselines but also suggests the potential for further improvement with some more advanced AE architecture.

**Qualitative Evaluation:** We qualitatively evaluate the latent cluster prototypes found by **DCAM** in Fig. 2 for Fashion MNIST (10 clusters) and Caltech Birds (10 out of 200 clusters). The figure shows the decoded prototypes or cluster centers, i.e., $\mathbf{d}(\rho_i)$, as well as their corresponding decoded closest and farthest cluster members ( as measured in the latent space) from the centers. Generally, the prototypes $\rho_i$ form an average image that matches the closest images well. The farthest cluster members still appear similar to their prototypes in most cases, with some exceptions: (i) In the 7th row for FMNIST an image that looks like a pant is grouped with dresses though the overall image shape is still similar. (ii) In the 5th row for CBird, the memory and the closest image are very similar but the farthest image appears significantly different.

Figure 2: Decoded images for the cluster prototypes (leftmost column in block) and the corresponding closest (center column in block) and farthest (right column in block) members in each cluster for Fashion MNIST (left block) and Caltech Birds (right block).

In addition to the above, we discuss our thorough empirical evaluation in Appendix C, reporting various clustering metrics, visualizing the evolution of the latent memories (cluster centers), studying the impact of latent dimensionality, and further details of selecting the best results via Pareto analysis along the SC and RL axes.

## 5 Discussion

We introduce a fresh integration of associative memories within an innovative deep clustering algorithm **DCAM** that leverages the energy-based attractor dynamics in latent space. Our findings demonstrate that **DCAM** significantly surpasses standard prototype-based and existing deep clustering methods. Our future work aims to extend it to multimodal deep clustering. Leveraging its flexibility to add other encoder/decoder frameworks with **DCAM**, we aim to explore transformer-based AE approaches. Additionally, we plan to explore how the energy landscape may help estimate the number of clusters directly from the data.

## REFERENCES


Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283, 2016.

Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, Maximilian Strobel, and Daniel Cremers. Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv:1801.07648*, 2018.

Daniel J Amit, Hanoch Gutfreund, and Haim Sompolinsky. Storing infinite numbers of patterns in a spin-glass model of neural networks. *Physical Review Letters*, 55(14):1530, 1985.

Ira Assent. Clustering high dimensional data. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(4):340–350, 2012.

Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 37–49. JMLR Workshop and Conference Proceedings, 2012.

Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *Machine learning for data science handbook: data mining and knowledge discovery handbook*, pp. 353–374, 2023.

Berthold Bein. Entropy. *Best Practice & Research Clinical Anaesthesiology*, 20(1):101–109, 2006.

Jinyu Cai, Shiping Wang, Chaoyang Xu, and Wenzhong Guo. Unsupervised deep clustering via contractive feature representation and focal loss. *Pattern Recognition*, 123:108386, 2022.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 132–149, 2018.

Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *Proceedings of the IEEE international conference on computer vision*, pp. 5879–5887, 2017.

Shlomo E Chazan, Sharon Gannot, and Jacob Goldberger. Deep clustering based on a mixture of autoencoders. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6. IEEE, 2019.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.

Zhiyuan Dang, Cheng Deng, Xu Yang, Kun Wei, and Heng Huang. Nearest neighbor matching for deep clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13693–13702, 2021.

Mete Demircigil, Judith Heusel, Matthias Löwe, Sven Upgang, and Franck Vermet. On a model of associative memory with huge storage capacity. *Journal of Statistical Physics*, 168(2):288–299, 2017.

Wengang Guo, Kaiyan Lin, and Wei Ye. Deep embedded k-means clustering. In *2021 International Conference on Data Mining Workshops (ICDMW)*, pp. 686–694. IEEE, 2021.

Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *Ijcai*, pp. 1753–1759, 2017a.

Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep clustering with convolutional autoencoders. In *International conference on neural information processing*, pp. 373–382. Springer, 2017b.

Benjamin Hoover, Yuchen Liang, Bao Pham, Rameswar Panda, Hendrik Strobelt, Duen Horng Chau, Mohammed Zaki, and Dmitry Krotov. Energy transformer. *Advances in Neural Information Processing Systems*, 36, 2024.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Xuan Huang, Zhenlong Hu, and Lin Lin. Deep clustering based on embedded auto-encoder. *Soft Computing*, 27(2):1075–1090, 2023.

Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994.

Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.

Mahmut Kaya and Hasan Şakir Bilge. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, ON, Canada, 2009.

Dmitry Krotov and John J Hopfield. Dense associative memory for pattern recognition. *Advances in neural information processing systems*, 29, 2016.

Dmitry Krotov and John J Hopfield. Large associative memory problem in neurobiology and machine learning. In *International Conference on Learning Representations*, 2021.

Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2): 129–137, 1982.

Carlo Lucibello and Marc Mézard. The exponential capacity of dense associative memories. *arXiv preprint arXiv:2304.14964*, 2023.

J MacQueen. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pp. 281–297, 1967.

ROBERTJ McEliece, Edwardc Posner, EUGENER Rodemich, and SANTOSHS Venkatesh. The capacity of the hopfield associative memory. *IEEE transactions on Information Theory*, 33(4): 461–482, 1987.

Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011.

Amin G. Oskouei, Mohammad A. Balafar, and Cina Motamed. Edcwrn: efficient deep clustering with the weight of representations and the help of neighbors. *Applied Intelligence*, 53(5):5845–5867, 2023.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.

Yazhou Ren, Jingyu Pu, Zhimeng Yang, Jie Xu, Guofeng Li, Xiaorong Pu, S Yu Philip, and Lifang He. Deep clustering: A comprehensive survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2024. URL `https://ieeexplore.ieee.org/abstract/document/10585323`.

Meitar Ronen, Shahaf E Finder, and Oren Freifeld. Deepdpm: Deep clustering with an unknown number of clusters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9861–9870, 2022.

Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning internal representations by error propagation, 1985.

Bishwajit Saha, Dmitry Krotov, Mohammed J Zaki, and Parikshit Ram. End-to-end differentiable clustering with associative memories. *arXiv preprint arXiv:2306.03209*, 2023.

Claude Sammut and Geoffrey I. Webb (eds.). *TF–IDF*, pp. 986–987. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_832. URL https://doi.org/10.1007/978-0-387-30164-8_832.

Michael Steinbach, Levent Ertöz, and Vipin Kumar. The challenges of clustering high dimensional data. In *New directions in statistical physics: econophysics, bioinformatics, and pattern recognition*, pp. 273–309. Springer, 2004.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6309–6318, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels. In *European conference on computer vision*, pp. 268–285. Springer, 2020.

Michel Verleysen and Damien François. The curse of dimensionality in data mining and time series prediction. In *International work-conference on artificial neural networks*, pp. 758–770. Springer, 2005.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th annual international conference on machine learning*, pp. 1073–1080, 2009.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

Chathurika S Wickramasinghe, Daniel L Marino, and Milos Manic. Resnet autoencoders for unsupervised feature learning from high-dimensional data: Deep models resistant to performance degradation. *IEEE Access*, 9:40511–40520, 2021.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pp. 478–487. PMLR, 2016.

Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2:165–193, 2015.

Mohammed J Zaki and Wagner Meira Jr. *Data mining and machine learning: Fundamental concepts and algorithms*. Cambridge University Press, 2020.

Sheng Zhou, Hongjia Xu, Zhuonan Zheng, Jiawei Chen, Zhao Li, Jiajun Bu, Jia Wu, Xin Wang, Wenwu Zhu, and Martin Ester. A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions. *ACM Comput. Surv.*, 57(3), November 2024. ISSN 0360-0300. doi: 10.1145/3689036. URL https://doi.org/10.1145/3689036.

## A Related Work

Clustering is a long-studied and well-reviewed problem in computer science, with various formulations and several applications (Kaufman & Rousseeuw, 2009; Zaki & Meira Jr, 2020). Given the success of deep learning, deep clustering has also attracted attention over the past decade (Ren et al., 2024; Aljalbout et al., 2018; Zhou et al., 2024). Inspired by t-SNE (Van der Maaten & Hinton, 2008), Xie et al. (2016) introduced DEC, enhancing clustering and feature representation by minimizing the Kullback-Leibler Divergence (KLD) to an auxiliary target distribution. However, a drawback is abandoning the decoder layer after pre-training, impacting the embedded space and clustering performance. Guo et al. (2017a) showed that keeping the decoder layer improves clustering (IDEC), and Guo et al. (2017b) proposed DCEC using convolutional autoencoders (CAE). Chazan et al. (2019) proposed DAMIC, a mixture of autoencoders for clustering, determined by minimizing the reconstruction loss without needing a regularization term. However, they leverage multiple AEs in their model, while we focus on schemes using a single AE. Huang et al. (2023) introduced an innovative embedded autoencoder architecture by incorporating it into both the encoding and decoding units of the outer autoencoder. Guo et al. (2021) proposed DEKM which works on the embedding space (after pretraining) and transforms it to a new cluster-friendly space using an orthonormal transformation matrix. However, discarding the decoder after pretraining for both of these methods may lead to the distortion of the embedded space, consequently hurting clustering performance. In addressing the automatic inference of the number of clusters in a dataset, Ronen et al. (2022) introduced DeepDPM. They proposed a novel loss inspired by EM in the Bayesian Gaussian Mixture Model framework, facilitating a new amortized inference in mixture models. It is worth noting that DeepDPM diverges from the typical encoder-decoder architecture, opting instead for a multilayer perceptron model.

While many deep clustering methods utilize KLD as a clustering objective, it falls short in preserving the global data structure (e.g., only within-cluster distances are prioritized, leaving uncertainties regarding between-cluster similarities), leading Oskouei et al. (2023) (EDCWRN) to advocate for cross-entropy over KLD. They incorporate feature weighting to emphasize essential features for clustering and employ a neighborhood technique to encourage similar representations for samples within the same cluster. Addressing another challenge with KLD regarding the presence of hard, misclassified samples, Cai et al. (2022) introduced focal loss to enhance label assignment in deep clustering methods and improved the representation learning module with a contractive penalty term, capturing more discriminative representations. However, it could lead to unintentional bias in the optimization focus between the representation learning and clustering modules. Dang et al. (2021) introduce a novel deep clustering framework (NNM) based on a two-level nearest neighbors matching approach. Distinguishing itself from prior methods (Van Gansbeke et al., 2020), NNM incorporates matching at both local and global levels, resulting in a notable enhancement in clustering performance. It also leverages SimCLR (Chen et al., 2020) to pretrain a representation learning model using the state-of-the-art contrastive learning loss. Our **DCAM** approach can flexibly incorporate various autoencoder architectures by leveraging the capabilities of associative memories, and can benefit from various architectural and pretraining advancements.

Associative Memories store multidimensional vectors as fixed point attractor states in a recurrent dynamical system. AMs form associations between the initial state and a final state (memory), creating disjoint basins of attractions which are crucial for clustering. A prominent example of AM is the classical Hopfield Network (Hopfield, 1982). It exhibits limited memory capacity, approximately storing only $\approx 0.14d$ arbitrary memories in a $d$ dimensional data domain (McEliece et al., 1987; Amit et al., 1985). Subsequently, Krotov & Hopfield (2016) proposed Dense Associative Memory (Dense AM) or Modern Hopfield Network introducing rapidly growing non-linearities (activation functions) into the system. This innovation allows for a denser arrangement of memories and achieves super-linear (in $d$) memory capacity (Demircigil et al., 2017; Lucibello & Mézard, 2023). With softmax activation, Dense AMs are closely related to the attention mechanism used in transformers (Ramsauer et al., 2020; Krotov & Hopfield, 2021; Hoover et al., 2024).

Recently, Saha et al. (2023) introduced `ClAM`, an end-to-end differentiable clustering approach, utilizing Dense AMs for clustering. However, there are other fundamental difference between how **DCAM** uses AMs versus `ClAM`, which performs clustering only in the ambient space utilizing AMs but there is no representation learning involved. In contrast, **DCAM** focuses on clustering in latent space, utilizing AMs to find good clusters and yet retain good reconstruction (which reflects the quality of the latent representations). In `ClAM`, AM is utilized to act as a differentiable argmin

solver for the $k$-means objective. In contrast, in **DCAM**, which involves representation learning, AM energy dynamics explicitly creates basins of attraction in the latent space, and moves/pushes the latent representations of the points into these basins, thereby explicitly inducing a clustered data distribution in the latent space. While the encoder is moving points into basins of attraction, the **DCAM** loss tries to minimize the information loss in the latent representations by having the decoder reconstruct these relocated latent representations. Finally, we show empirically that when `ClAM` is directly applied in the latent space learned by a pretrained autoencoder, it does not yield competitive clustering performance. Our novel **DCAM** approach, which is inherently a deep clustering method that jointly clusters and learns effective latent representations, yields much better performance.

*To our knowledge, the coupling of deep clustering with energy dynamics in latent space as done in **DCAM** for cluster-guided latent space learning has not been considered in the literature before.* **DCAM** continuously refines both the encoder and decoder networks and at the same time integrate the AM learning dynamics to cluster the points into $k$ groups. This bears semblance to vector-quantized variational AEs (van den Oord et al., 2017), where the task is to learn a discrete vector code for each point. However, this assignment is non-differentiable, requiring gradient approximation, and there is no clustering objective considered. Also related is the task of deep metric learning (Kaya & Bilge, 2019), where the task is to learn a distance function between samples in latent space. However, this requires the use of labeled data for full or weak supervision.

### A.1 **DCAM** ALGORITHM

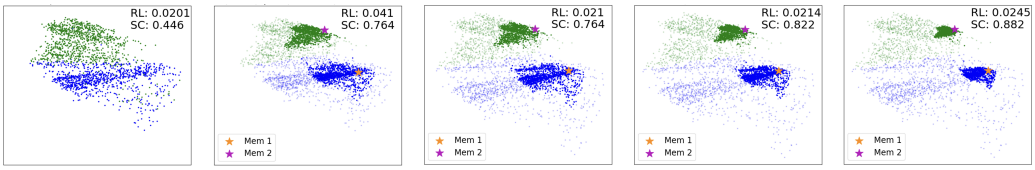
**Algorithm 1:** **DCAM** Algorithm

**Train**$(S, k, N, T, \epsilon_{\mathbf{e}}, \epsilon_{\mathbf{d}}, \epsilon_{\boldsymbol{\rho}}, \gamma)$
- Pretrain $(\mathbf{e}, \mathbf{d})$ as autoencoder, minimizing $\mathcal{L}_r(\mathbf{e}, \mathbf{d})$
- $\boldsymbol{\rho} \leftarrow \{\mathbf{e}(x), x \in M\}$, $M$ are random $k$ samples from S
- **for** *epoch* $n = 1, \ldots, N$ **do**
  - **for** *batch* $B \in S$ **do**
    - Batch loss $\bar{\mathcal{L}} \leftarrow 0$
    - **for** *example* $x \in B$ **do**
      - $v \leftarrow \mathbf{e}(x)$ `//encode`
      - $v' \leftarrow A^T_{\boldsymbol{\rho}}(v)$ `//energy descent`
      - $\bar{\ell} \leftarrow \|x - \mathbf{d}(v')\|^2$ `//loss`
      - $\bar{\mathcal{L}} \leftarrow \bar{\mathcal{L}} + \bar{\ell}$
    - $\rho_i \leftarrow \rho_i - \epsilon_{\boldsymbol{\rho}} \nabla_{\rho_i} \bar{\mathcal{L}},\ \forall i \in [\![k]\!]$
    - $\mathbf{e} \leftarrow \mathbf{e} - \epsilon_{\mathbf{e}} \nabla_{\mathbf{e}} \bar{\mathcal{L}}$
    - $\mathbf{d} \leftarrow \mathbf{d} - \epsilon_{\mathbf{d}} \nabla_{\mathbf{d}} \bar{\mathcal{L}}$
- **return** $\mathbf{e}, \mathbf{d}, \boldsymbol{\rho}$

**Infer**$(S, \mathbf{e}, \mathbf{d}, \boldsymbol{\rho})$
- Cluster assignments $C \leftarrow \emptyset$
- **for** $x \in S$ **do**
  - $v' \leftarrow A^T_{\boldsymbol{\rho}}(\mathbf{e}(x))$
  - $C \leftarrow C \cup \left\{ \arg\min_{i \in [\![k]\!]} \|\rho_i - v'\|^2 \right\}$
- **return** *Per-point cluster assignments* $C$

Alg. 1 shows the pseudo-code for **DCAM**. It first pretrains encoder $\mathbf{e}$ and decoder $\mathbf{d}$, and starts from $k$ random prototypes $\boldsymbol{\rho}$. The cluster assignment is done with $T$ recursion of the AM attractor dynamics operator $A_{\boldsymbol{\rho}}$ parameterized with the centers $\boldsymbol{\rho} = \{\rho_i, i \in [\![k]\!]\}$. The per-sample loss $\bar{\ell}$ of **DCAM** (line 10) is added to the batch loss. We optimize for $N$ epochs via gradient descent, with learning rates $\{\epsilon_{\mathbf{e}}, \epsilon_{\mathbf{d}}, \epsilon_{\boldsymbol{\rho}}\}$ for $\mathbf{e}, \mathbf{d}, \boldsymbol{\rho}$ respectively. Upon solving Eq. (2), we obtain a trained encoder and decoder, and memories in the latent space, and we can utilize them to obtain the final partition the data (see the **Infer** subroutine in Alg. 1). Fig. 3 shows an illustration of how **DCAM** evolves over the epochs by minimizing our loss (Eq. (2)) while successively finding better clusters.

**DCAM** provides various advantages over previous deep clustering formulations: (i) Our novel per-sample loss $\bar{\ell}(x, \mathbf{e}, \mathbf{d}, \boldsymbol{\rho})$ does not involve a separate clustering loss thus obviating the need for the balancing hyperparameter $\gamma$. (ii) The updates for all the parameters in **DCAM** are more explicitly




(a) Pretrained (b) $E$=1, $T$=7 (c) $E$=200, $T$=8 (d) $E$=400, $T$=11 (e) $E$=800, $T$=15

Figure 3: **Clustering with DCAM**. Reconstruction loss (RL; refers to the loss $\bar{\mathcal{L}}$ in Eq. (2)) and clustering quality (SC; refers to Silhouette Coefficient) for different epochs ($E$) and number of AM steps ($T$) for Fashion-MNIST dataset with two clusters, where Fig. 3a represents the pretrained latent representations. The light (faded) colors indicate the encoded points $v$ before applying the attractor dynamics, whereas dark colors indicate the points $v' = A_{\boldsymbol{\rho}}^T(v)$ after that step. The colored stars represent the learned prototypes. **DCAM** discovers more compact and clustering-friendly latent representations that simultaneously have higher clustering and reconstruction quality.

tied together with the $\mathbf{d} \circ A_{\boldsymbol{\rho}}^T \circ \mathbf{e}$ composition in the $\mathbf{d}(A_{\boldsymbol{\rho}}^T(\mathbf{e}(x)))$ term. *This ties the representation learning and clustering objectives more intricately.* (iii) **DCAM** continues to have all the advantages of traditional deep clustering, being end-to-end differentiable since all operators in the above composition are differentiable, and so is the discrete cluster center assignment via $T$ recursions of the attractor dynamics operator $A_{\boldsymbol{\rho}}$. (iv) It is architecture agnostic – we can select a problem dependent encoder and decoder. For example, convolutional or residual networks for images or fully-connected feed-forward networks for text or tabular data. In essence, **DCAM** introduces an inductive bias over the latent space via AM, which defines an energy function and utilizes the attractor dynamics to help cluster the points.

# B Experimental Details

## B.1 Evaluation Metrics:

A common metric to evaluate and benchmark deep clustering algorithms is by computing the overlap between the clusters in the latent space and the partitions obtained from some ground-truth labels, e.g., Normalized Mutual Information (NMI) (Vinh et al., 2009). Nevertheless, *it is critical to ensure that NMI (or any other label-dependent metric) is not utilized for hyperparameter selection* since that leaks supervision into the unsupervised task of deep clustering. Unfortunately, for many of reported results, it is not clear how hyperparameters are selected without being influenced at NMI (since they simply report results with the highest NMI). Furthermore, existing works typically report NMI without explicitly discussing reconstruction loss, which may not align with the primary goals of deep clustering.

Given the unsupervised nature of the deep clustering, hyperparameters should be selected based on unsupervised metrics that do not utilize ground-truth labels to evaluate clustering quality. Thus, we report results on optimizing for the unsupervised Silhouette Coefficient (SC) (Rousseeuw, 1987) metric, while keeping the reconstruction loss (RL) below some user-defined threshold. We do also report NMI results in Appendix C.2, along with other metrics.

## B.2 Dataset Details

To evaluate **DCAM**, we conducted our experiments on eight standard benchmark datasets, including USPS (Hull, 1994), Fashion-MNIST (Xiao et al., 2017), CIFAR-10 and CIFAR-100 (Krizhevsky, 2009), STL-10 (Coates et al., 2011), Caltech_birds2010 (Welinder et al., 2010), 20-NG from Sklearn and Reuters-10k from TensorFlow datasets. The later two are text datasets, whereas the others are image datasets. For both text datasets, we calculate TFIDF (Sammut & Webb, 2010) features based on the 2000 most frequent words, following a similar approach as Oskouei et al. (2023). However, we consider the original number of categories as the true number of clusters, which is 46 for Reuters-10k and 20 for 20-NG. For Caltech_birds2010, as there are images of various shapes, we resize all images

to (128, 128, 3) for uniformity and ease of implementation. Table 2 provides the statistics for the datasets used in our experiments.

Table 2: Descriptions of datasets

| Dataset | Short name | # Points | Shape | # Classes | # Type |
|---|---|---|---|---|---|
| Fashion MNIST | FM | 60000 | (28, 28, 1) | 10 | Image |
| CIFAR-10 | C-10 | 50000 | (32, 32, 3) | 10 | Image |
| CIFAR-100 | C-100 | 50000 | (32, 32, 3) | 100 | Image |
| USPS | USPS | 2007 | (16, 16, 1) | 10 | Image |
| STL-10 | STL | 5000 | (96, 96, 3) | 10 | Image |
| Caltech_birds2010 | CBird | 3000 | (128, 128, 3) | 200 | Image |
| Reuters-10k | R-10k | 11228 | 2000 | 46 | Text |
| 20-NG | 20NG | 18846 | 2000 | 20 | Text |

### B.3 METRICS USED

To assess the performance of **DCAM**, we utilize the Silhouette Coefficient (SC) (Rousseeuw, 1987) as an unsupervised metric for measuring clustering quality. SC scores range from $-1$ to 1, where 1 indicates perfect clustering and $-1$ indicates completely incorrect labels. A score close to 0 suggests the existence of overlapping clusters. We also employ Normalized Mutual Information (NMI) (Vinh et al., 2009) to evaluate the alignment between the partition obtained by **DCAM** and the ground truth clustering labels. NMI scores range from 0 (completely incorrect) to 1 (perfect clustering). Additionally, we compute Reconstruction Loss (RL), representing the mean squared error between original and reconstructed points, where lower is better. Entropy (ETP) (Bein, 2006) and Cluster Size (CS) are computed to assess cluster balance. In clustering, higher entropy (the highest value is $\log_2(k)$ for each dataset, where $k$ is the number of true clusters) indicates more balanced clusters, while lower values suggest potential imbalance, possibly involving singleton or very small clusters. Entropy ($H(X)$) is calculated based on the distribution of data points across clusters:

$$H(X) = -\sum_{i=1}^{k} P(C_i)\ \log_2 P(C_i)$$

where, $P(C_i)$ is the proportion of data points in cluster $C_i$ relative to the total number of data points. Cluster Size (CS) indicates the largest and smallest clusters (in terms of the number of data points) identified in the dataset (more balanced clustering is better).

### B.4 PARAMETER SETTINGS

For Convolutional AE or CAE, for $k$-means, Agglomerative, `ClAM`, DCEC, DEKM, and **DCAM**, we adopt an architecture identical to DCEC. The encoder network structure follows $\text{conv}^5_{32} \rightarrow \text{conv}^5_{64} \rightarrow \text{conv}^3_{128} \rightarrow \text{FC}_d$, where $\text{conv}^k_n$ represents a convolutional layer with $n$ filters and a kernel size of $k \times k$, and $\text{FC}_d$ denotes a fully connected layer of dimension $d$. Here, $d$ is the number of true clusters in the dataset, and serves as the latent dimension. The decoder mirrors the encoder.

The ResNet AE or RAE approach draws inspiration from the standard ResNet block described by Wickramasinghe et al. (2021). For DCEC, DEKM, and **DCAM** a streamlined configuration is employed using two filters with sizes 32 and 64. The size of the embedded representation is maintained at $d$, corresponding to the number of clusters in the dataset, as in the previous setup. In this experiment, the number of repeating layers in the ResNet block is set to 2. To enhance model performance, batch normalization and leakyReLU are incorporated. For a given number of repeats ($f$), the total number of hidden layers is calculated as 2 + ($f$ * number of filters), resulting in 6 layers in our case.

The EDCWRN AE or EAE, is that from Oskouei et al. (2023), so for both EDC and **DCAM**, we follow the proposed architecture, where the encoder network is configured as a fully connected multilayer perceptron (MLP) with dimensions $i$-500-500-2000-$d$ for all datasets, where $i$ represents

the dimension of the input space (features), and $d$ is the number of clusters in the dataset. Similarly, the decoder network mirrors the encoder, constituting an MLP with dimensions $d$-2000-500-500-$i$. All internal layers, except for the input, output, and embedding layers, use the ReLU activation function.

All three architectures described above are pretrained end-to-end for 100 epochs using Adam (Kingma & Ba, 2014) with default parameters. The number of clusters $k$ is not a hyperparameter, but rather is taken as the true number of classes in each dataset. Also, as noted above, we set the latent dimensionality $d$ (or $m$) the same as the number of true classes $k$ in the dataset, i.e., $m = d = k$.

### B.5 Implementation details

We conduct a comparative analysis of **DCAM** against established clustering methods, including $k$-means (Lloyd, 1982), agglomerative clustering (or Agglo.) (Müllner, 2011), `ClAM` (Saha et al., 2023), DCEC (Guo et al., 2017b), DEKM (Guo et al., 2021) and EDCWRN (or EDC) Oskouei et al. (2023). We evaluate $k$-means, agglomerative clustering, and `ClAM` in the ambient space (denoted as NAE) and in the latent space obtained through a pretrained Convolutional Autoencoder (CAE) as used in DCEC (Guo et al., 2017b). For DCEC amd DEKM, we consider a ResNet-based AE (RAE) (Wickramasinghe et al., 2021) along with their original CAE. For **DCAM**, we extend our exploration to include not only the CAE and RAE architectures but also EDCWRN-based (Oskouei et al., 2023) Autoencoder (EAE) (originally proposed by Guo et al. (2017a)) to analyze its impact on the algorithm. We also compare **DCAM** with state-of-the-art SimCLR (Chen et al., 2020) based (contrastive learning) SCAN (Van Gansbeke et al., 2020) and NNM (Dang et al., 2021) deep clustering schemes.

We implement and evaluate **DCAM** using the Tensorflow (Abadi et al., 2016) library while employing `scikit-learn` (Pedregosa et al., 2011) for clustering quality metrics. We train our models on a single node with 1 NVIDIA RTX A6000 (48GB RAM) and a 16-core 2.4GHz Intel Xeon(R) Silver 4314 CPU. Hyperparameters are tuned individually for each dataset to maximize the Silhouette Coefficient (Rousseeuw, 1987). Table 5 lists the hyperparameters, their roles, and respective values/ranges.

For baseline schemes like $k$-means and agglomerative, we use the `scikit-learn` library implementation, adjusting hyperparameters for optimal performance on each dataset. For DCEC (Guo et al., 2017b) and DEKM (Guo et al., 2021), we leverage their Tensorflow implementation[1] [2], for EDCWRN (Oskouei et al., 2023), we utilze their Python implementation[3], and for `ClAM` (Saha et al., 2023) we use their Tensorflow implementation[4]. Likewise we use the author provide implementations for SCAN (Van Gansbeke et al., 2020) [5] and NNM (Dang et al., 2021)[6].

Table 3: Per-method best RRL across all architectures (while SC is within 10% of the best SC of the method).

| Dataset | SC | | | | | | | | | RRL | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $k$-means | Agglo. | ClAM | DCEC | DEKM | EDC | SCAN | NNM | DCAM | DCEC | DEKM | EDC | DCAM |
| FM | 0.257 | 0.201 | 0.279 | 0.898 | 0.785 | 0.521 | - | - | **0.922** | **9.8**$^{\blacktriangledown}$ | 321 | 143 | 42.2 |
| C-10 | 0.084 | 0.372 | 0.208 | 0.786 | 0.622 | 0.541 | 0.541 | 0.587 | **0.809** | 0.9$^{\blacktriangledown}$ | 180 | 74.3 | **20.4**$^{\blacktriangledown}$ |
| C-100 | 0.015 | 0.149 | 0.053 | 0.572 | 0.047 | 0.337 | 0.321 | 0.358 | **0.921** | **18.6** | 870 | 33.3 | 27.5 |
| USPS | 0.195 | 0.158 | 0.194 | **0.929** | 0.843 | 0.491 | - | - | 0.914 | **26.3**$^{\blacktriangledown}$ | 2326 | 40 | 8.7 |
| STL | 0.079 | 0.270 | 0.108 | 0.812 | 0.804 | 0.431 | 0.552 | 0.540 | **0.923** | 79.2 | 234 | 155 | **27.7** |
| CBird | -0.019 | 0.094 | -0.026 | 0.282 | 0.018 | 0.188 | - | - | **0.413** | 286 | 1036 | 102 | **1.8** |
| R-10k | -0.010 | 0.114 | -0.002 | - | - | 0.035 | - | - | **0.673** | - | - | **60** | 120 |
| 20NG | -0.021 | 0.114 | -0.008 | - | - | 0.099 | - | - | **0.287** | - | - | **25**$^{\blacktriangledown}$ | 50 |

[1] https://github.com/XifengGuo/DCEC

[2] https://github.com/spdj2271/DEKM/blob/main/DEKM.py

[3] https://github.com/Amin-Golzari-Oskouei/EDICWRN

[4] https://github.com/bsaha205/clam

[5] https://github.com/wvangansbeke/Unsupervised-Classification

[6] https://github.com/ZhiyuanDang/NNM

Table 4: SC for image datasets, comparing **DCAM** to baselines with different encoder/decoder architectures. Best for each dataset is in bold. See text for details. *Higher is better.*

| Dataset | Convolutional AE | | | ResNet AE | | | EAE | |
|---|---|---|---|---|---|---|---|---|
| | DCEC | DEKM | **DCAM** | DCEC | DEKM | **DCAM** | EDC | **DCAM** |
| FM | 0.923 | 0.785 | **0.970** | 0.824 | 0.742 | 0.922 | 0.521 | 0.715 |
| C-10 | 0.787 | 0.622 | **0.863** | 0.667 | 0.461 | 0.697 | 0.541 | 0.731 |
| C-100 | 0.572 | 0.047 | 0.598 | 0.557 | 0.036 | **0.921** | 0.337 | 0.636 |
| USPS | **0.935** | 0.882 | 0.914 | 0.909 | 0.843 | 0.914 | 0.491 | 0.911 |
| STL | 0.766 | 0.745 | 0.919 | 0.812 | 0.804 | 0.865 | 0.431 | **0.923** |
| CBird | 0.386 | 0.018 | **0.448** | 0.282 | 0.035 | 0.377 | 0.188 | 0.446 |

### B.6 HYPERPARAMETERS FOR **DCAM**

Table 5: Hyperparameters, their roles and range of values for **DCAM**.

| **Hyperparameter** | **Used Values** |
|---|---|
| Inverse temperature, $\beta$ | $[10^{-5}, ..., 5]$ |
| Batch size | $[16, 32, 64, 128, 256]$ |
| Initial learning rate (AM), $\epsilon_{am}$ | $[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}]$ |
| Initial learning rate (AE), $\epsilon_{ae}$ | $[10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}]$ |
| Reduce LR patience (epochs) | $[5, 10, 15]$ |

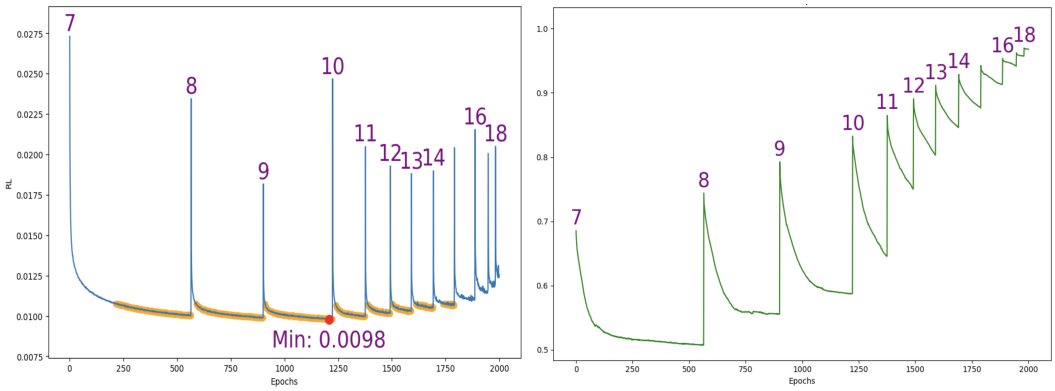
We extensively tune all hyperparameters (Table 5) for the optimal results in **DCAM**. We found that the inverse temperature $\beta$ serves as the most critical hyperparameter, which we explore in the range of $[10^{-5}, ..., 5]$ for tuning. We employ curriculum learning for the number of AM steps $T$ from the data, where $T$ starts with a small value (e.g., 1) and increases up to 20 based on the reconstruction loss. Fig. 4 visualizes this curriculum learning for $T$ for FMNIST dataset where $T$ starts with 7 and ends at 18. We employ the Adam optimizer while keeping separate initial learning rates for the AM and AE networks. If the training loss does not improve for a certain number of epochs, we decrease the learning rate by a factor of 0.8. We do this for a certain patience value (curriculum patience) after that we increase $T$ by 1. This process continues until the training loss reaches a minimum threshold ($10^{-9}$) or $T$ reaches its maximum value (20). In Fig. 4, the red point indicates the lowest training loss (reconstruction error) and orange points indicate the reconstruction losses within 10% of the lowest reconstruction loss. We can see that $T = 14$ has reconstruction loss within 10% of the lowest reconstruction loss and a high value of silhouette coefficient ($> 0.9$). In this case, we select 14 as the optimal value of $T$ as it has a good trade-off between the reconstruction loss and silhouette coefficient. By doing curriculum learning, we avoid treating $T$ as a separate hyperparameter. The best hyperparameter values for various datasets for **DCAM** are detailed in Table 6.

Table 6: Best hyperparameters for different datasets for **DCAM**. '-' denotes NA.

| **Dataset** | **Inverse temperature, $\beta$** | | | **Layers, $T$** | | | **Batch size** | | | **Learning rate(AM)** | | | **Learning rate (e)** | | | **Learning rate (d)** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CAE | RAE | EAE | CAE | RAE | EAE | CAE | RAE | EAE | CAE | RAE | EAE | CAE | RAE | EAE | CAE | RAE | EAE |
| FM | 0.5 | 0.09 | 0.7 | 15 | 15 | 10 | 64 | 64 | 64 | 0.001 | 0.001 | 0.1 | 0.0000001 | 0.0000001 | 0.0000001 | 0.001 | 0.001 | 0.001 |
| C-10 | 2 | 0.02 | 0.5 | 15 | 15 | 12 | 64 | 64 | 64 | 0.001 | 0.001 | 0.01 | 0.0000001 | 0.0000001 | 0.0000001 | 0.001 | 0.001 | 0.001 |
| C-100 | 1 | 0.005 | 5 | 10 | 10 | 10 | 64 | 64 | 64 | 0.001 | 0.001 | 0.001 | 0.0000001 | 0.0000001 | 0.0000001 | 0.001 | 0.001 | 0.001 |
| USPS | 0.5 | 1 | 1 | 15 | 10 | 15 | 64 | 64 | 32 | 0.01 | 0.01 | 0.1 | 0.0000001 | 0.0000001 | 0.0000001 | 0.001 | 0.001 | 0.01 |
| STL | 0.5 | 0.003 | 0.1 | 15 | 10 | 12 | 64 | 64 | 128 | 0.001 | 0.01 | 0.1 | 0.0000001 | 0.0000001 | 0.0000001 | 0.001 | 0.001 | 0.0001 |
| CBird | 0.05 | 0.00015 | 0.005 | 15 | 10 | 15 | 64 | 64 | 64 | 0.01 | 0.001 | 0.1 | 0.0000001 | 0.0000001 | 0.0000001 | 0.001 | 0.001 | 0.001 |
| R-10K | - | - | 10 | - | - | 10 | - | - | 64 | - | - | 0.01 | - | - | 0.0000001 | - | - | 0.1 |
| 20-NG | - | - | 1.5 | - | - | 15 | - | - | 64 | - | - | 0.1 | - | - | 0.0000001 | - | - | 0.1 |

### B.7 HYPERPARAMETERS FOR BASELINES

We compare **DCAM** with baseline clustering schemes $k$-means and agglomerative from `scikit-learn`, `ClAM`, DCEC, DEKM and EDCWRN. For $k$-means and agglomerative, we




(a) RL vs Training

(b) SC vs Training

Figure 4: Reconstruction loss (RL) and clustering quality (SC) for varying number of steps (T) for FMNIST. Red point in Fig. 4a indicates the lowest reconstruction loss and orange points indicate the reconstruction loss within 10% of this lowest reconstruction loss.

perform a comprehensive search for tuning different hyperparameters available in `scikit-learn` and pick the best results. For DCEC, DEKM and EDCWRN, we tuned all related hyperparams to obtain the best SC (for 10% RRL) and best RL (for 10% of best SC). For `ClAM` we precisely replicate the hyperparameter search criteria outlined in its respective paper, which closely aligns with our approach for **DCAM**, as detailed in Table 5. Table 7 provides a brief description of the hyperparameters and their roles in the baseline schemes.

# C Detailed and Additional Experimental Results

## C.1 Pretrained losses for all architecture and all datasets

Table 8 records the pretrained reconstruction losses (RL) for all architectures and all datasets. These are the base RL values $RL_p$ used when computing RRL.

## C.2 Detailed results with various clustering quality metrics

Table 9 provides a comprehensive overview of the metrics (SC, NMI, RL, ETP, and CS) for **DCAM**, and corresponding baselines, focusing on the best SC in each method across various AE architectures where RL is constrained to 10% of the pretrained AE loss. For $k$-means, Agglomerative and `ClAM`, we apply them both in the original ambient space (No-AE or NAE) and in the latent space (utilizing CAE). RL is not presented for $k$-means, Agglomerative and `ClAM` for the original space and for CAE as it remains consistent across the three methods after pretraining. Similarly, Table 10 provides a similar overview of the metrics (SC, NMI, RL, ETP, and CS) for **DCAM**, and corresponding baselines, focusing on the best Relative RL (RRL) in each method across various AE architectures where SC is constrained to 10% of the best/peak SC of the method. Table 11 represents all corresponding metrics focusing on the best NMI in each method. These tables highlight that **DCAM** exhibits strong performance not only in terms of SC and RL, but also when compared to the ground truth labels via NMI. In fact, for NMI, **DCAM** has the best values in 5 out of the 8 datasets (DCEC has the best values on the other 3). Additionally, **DCAM** clusters maintain reasonable entropy (ETP) and cluster size (CS), ensuring a balanced clustering outcome.

For an understanding of the importance of ETP and CS in clustering, consider the case of Agglomerative clustering in the latent space (CAE) on the CIFAR-10 dataset (see Table 9). In this instance, almost all points (49991 out of 50000) belong to one cluster, while the other 9 clusters contain only one data point each, indicating very poor clustering. The low entropy (0.003) further highlights the deficiency of the clustering.

In certain situations, when comparing two clustering methods, it can happen that a method performs better in terms of SC and RL but still exhibits a lower NMI compared to another method (see Table 9

Table 7: Hyperparameters (HPs), their roles and range of values for the baseline clustering schemes.

| Baseline | HP | Role | Used Values |
|---|---|---|---|
| $k$-means | init | Initialization method | ['k-means++', 'random'] |
| | n_init | Number of time the k-means algorithm will be run | 1000 |
| Agglomerative | affinity | Metric used to compute the linkage | ['euclidean', 'l1', 'l2', 'manhattan', 'cosine'] |
| | linkage | Linkage criterion to use | ['single', 'average', 'complete', 'ward'] |
| DCEC | batch_size | Size of each batch | [64, 128, 256] |
| | maxiter | Maximum number of iteration | [2e4, 3e4] |
| | alpha | Degree of freedom of student's t-distribution | 1 |
| | gamma | Coefficient of clustering loss | [0.01, 0.1, 1, 10] |
| | update_interval | Interval at which the predicted and target distributions are updated | [1, 2, 3, 4, 5, 10, 20, 30, 40, 50, 75, 100, 125, 140, 150, 200] |
| | tol | Tolerance rate | 0.001 |
| DEKM | batch_size | Size of each batch | [64, 128, 256] |
| | maxiter | Maximum number of iteration | 2e4 |
| | hidden_units | Number of latent dimension | Number of true cluster as per dataset |
| | update_interval | Interval at which the predicted and target distributions are updated | [10, 20, 30, 40, 50, 75, 100, 125, 140, 150, 200] |
| | tol | Tolerance rate | 0.000001 |
| EDCWRN | batch_size | Size of each batch | [64, 128, 256] |
| | maxiter_pretraining | Maximum number of iteration in pertaining | 500*batch_size |
| | maxiter_clustering | Maximum number of iteration in clustering | [8000, 16000, 24000] |
| | gamma | Coefficient of clustering loss | [0.01, 0.1, 1, 10] |
| | update_interval | Interval at which the predicted and target distributions are updated | [1, 2, 3, 4, 5, 10, 20, 30, 40, 50, 75, 100, 125, 140, 150, 200] |
| | tol | Tolerance rate | 0.0001 |
| ClAM | $\beta$ | Inverse temperature | [$10^{-5}$ - 5] |
| | $T = 1/\alpha = \tau/dt$ | Number of layers | [2-20] |
| | batch_size | Size of each batch | [8, 16, 32, 64, 128, 256] |
| | $\epsilon$ | Adam initial learning rate | [$10^{-4}$, $10^{-3}$, $10^{-2}$, $10^{-1}$] |
| | $\epsilon$_factor | Reduce LR by factor | 0.8 |
| | $\epsilon$_patience | Reduce LR patience (epochs) | 5 |
| | $\epsilon$_min | Minimum LR | $10^{-5}$ |
| | $\epsilon$_loss_threshold | Reduce LR loss threshold | $10^{-3}$ |
| | max_epochs | Maximum Number of epochs | 200 |
| | restart | Number of restart | 10 |
| | mask_prob | Mask probability | [0.1, 0.12, 0.15, 0.2, 0.25, 0.3] |
| | mask_val | Mask value | ['mean', 'min', 'max'] |

Table 8: Per-dataset, per-architecture pretrained loss. Note abbreviations Conv-AE→CAE, Resnet-AE→RAE, and EDCWRN-AE→EAE. Further, '-' denotes NA.

| Dataset | Architecture | | |
|---|---|---|---|
| | CAE | RAE | EAE |
| FM | 0.0122 | 0.0083 | 0.0087 |
| C-10 | 0.0220 | 0.0180 | 0.0167 |
| C-100 | 0.0070 | 0.0040 | 0.0096 |
| USPS | 0.0019 | 0.0023 | 0.0005 |
| STL | 0.0179 | 0.0173 | 0.0206 |
| CBird | 0.0055 | 0.0036 | 0.0187 |
| R-10K | - | - | 0.0010 |
| 20NG | - | - | 0.0008 |

for USPS where **DCAM** outperforms DCEC in both CAE and RAE architecture in both SC and RL, however, the NMI is worse than DCEC in both cases). This indicates that the alignment of semantic class (ground truth or true underlying structure) with the geometric characteristics of the data might not be consistent or straightforward.

Table 9: **Metrics obtained by DCAM and baselines corresponding to the best SC (RL within 10% of the pretrained AE loss).** The best performance for each dataset is in **boldface**. (note abbreviations DCEC→DC, EDCWRN→EDC, Entropy→ETP, Cluster-size→CS, No-AE→NAE, Conv-AE→CAE, EDCWRN-AE→EAE, Resnet-AE→RAE). '-' denotes NA. x▼ indicates negative RRL which means the RL of the method is x% less than the pretrained AE loss.

| Data | Met | Kmeans | | Agglo | | C1AM | | DC | | DEKM | | EDC | DCAM | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | NAE | CAE | NAE | CAE | NAE | CAE | CAE | RAE | CAE | RAE | | CAE | EAE | RAE |
| FM | SC | 0.154 | 0.257 | 0.109 | 0.201 | 0.158 | 0.279 | 0.923 | 0.726 | 0.260 | 0.258 | 0.483 | **0.970** | 0.663 | 0.712 |
| | NMI | 0.511 | 0.643 | 0.534 | 0.624 | 0.521 | 0.622 | 0.558 | 0.624 | 0.551 | 0.609 | 0.495 | 0.472 | 0.511 | 0.488 |
| | RL | - | 0.0122 | - | 0.0122 | - | 0.0122 | 0.0134 | 0.0091 | 0.0105 | 0.0097 | 0.0096 | 0.0120 | 0.0096 | 0.0091 |
| | RRL | - | 0.0 | - | 0.0 | - | 0.0 | 9.8 | 9.6 | **13.9**▼ | 18.1 | 10 | 1.6▼ | 10 | 9.6 |
| | ETP | 3.17 | 3.17 | 3.14 | 3.2 | 2.81 | 2.80 | 3.23 | 3.23 | 3.14 | 3.15 | 3.11 | 2.83 | 3.14 | 3.11 |
| | CS | 9617-2361 | 11145-2744 | 11830-1860 | 10298-2544 | 19032-1524 | 15679-2 | 10208-2733 | 8914-3338 | 12360-2310 | 10974-2724 | 12118-1478 | 20448-504 | 11734-2251 | 15906-2082 |
| C-10 | SC | 0.050 | 0.084 | 0.158 | 0.372 | 0.073 | 0.208 | 0.787 | 0.659 | 0.116 | 0.082 | 0.511 | **0.863** | 0.632 | 0.697 |
| | NMI | 0.078 | 0.122 | 0.0005 | 0.0004 | 0.073 | 0.015 | 0.074 | 0.094 | 0.123 | 0.129 | 0.112 | 0.075 | 0.061 | 0.079 |
| | RL | - | 0.0220 | - | 0.0220 | - | 0.0220 | 0.0241 | 0.0197 | 0.0239 | 0.0199 | 0.0184 | 0.0178 | 0.0184 | 0.0170 |
| | RRL | - | 0.0 | - | 0.0 | - | 0.0 | 9.6 | 8.9 | 8.6 | 11.1 | 10 | **19.5**▼ | 10 | 5▼ |
| | ETP | 3.27 | 3.19 | 0.006 | 0.003 | 2.50 | 0.24 | 3.22 | 2.99 | 3.25 | 3.15 | 3.24 | 2.83 | 2.65 | 2.81 |
| | CS | 7105-2734 | 9779-2524 | 49979-1 | 49991-1 | 23544-582 | 48234-1 | 8511-2610 | 11229-1724 | 7905-3245 | 11731-2107 | 8198-2632 | 17521-425 | 13771-570 | 17121-569 |
| C-100 | SC | 0.015 | -0.020 | 0.028 | 0.149 | 0.018 | 0.053 | 0.314 | 0.470 | -0.007 | -0.016 | 0.311 | **0.598** | 0.536 | 0.482 |
| | NMI | 0.161 | 0.183 | 0.036 | 0.004 | 0.153 | 0.156 | 0.104 | 0.119 | 0.238 | 0.184 | 0.181 | 0.110 | 0.202 | 0.125 |
| | RL | - | 0.0070 | - | 0.0070 | - | 0.0070 | 0.0059 | 0.0043 | 0.0046 | 0.0041 | 0.0106 | 0.0069 | 0.0099 | 0.0044 |
| | RRL | - | 0.0 | - | 0.0 | - | 0.0 | 15.7▼ | 7.5 | **34.3**▼ | 2.5 | 4.3 | 1.4▼ | 3.1 | 10 |
| | ETP | 6.53 | 6.48 | 0.940 | 0.052 | 6.51 | 4.38 | 5.54 | 4.52 | 6.25 | 6.46 | 6.49 | 4.16 | 5.85 | 4.03 |
| | CS | 1160-129 | 1395-23 | 38814-1 | 49834-1 | 1317-177 | 13950-11 | 2255-325 | 12721-122 | 1715-5 | 1322-87 | 999-216 | 11085-112 | 4116-32 | 8245-112 |
| USPS | SC | 0.143 | 0.195 | 0.124 | 0.158 | 0.144 | 0.194 | **0.935** | 0.758 | 0.195 | 0.217 | 0.461 | 0.820 | 0.872 | 0.891 |
| | NMI | 0.573 | 0.628 | 0.627 | 0.680 | 0.475 | 0.619 | 0.732 | 0.761 | 0.631 | 0.665 | 0.467 | 0.444 | 0.347 | 0.428 |
| | RL | - | 0.0019 | - | 0.0019 | - | 0.0019 | 0.0018 | 0.0019 | 0.0020 | 0.0024 | 0.0005 | 0.0021 | 0.0006 | 0.0025 |
| | RRL | - | 0.0 | - | 0.0 | - | 0.0 | 5.3▼ | **17.4**▼ | 5.3 | 4.3 | 0.0 | 10 | 10 | 8.7 |
| | ETP | 3.27 | 3.23 | 3.26 | 3.27 | 3.10 | 3.16 | 3.26 | 3.29 | 3.23 | 3.25 | 3.29 | 3.12 | 2.78 | 2.99 |
| | CS | 284-121 | 359-89 | 333-121 | 328-104 | 420-53 | 375-64 | 287-106 | 281-138 | 379-90 | 319-99 | 295-134 | 442-71 | 841-76 | 524-49 |
| STL | SC | 0.039 | 0.079 | 0.158 | 0.270 | 0.051 | 0.108 | 0.132 | 0.259 | 0.082 | 0.081 | 0.411 | 0.475 | **0.891** | 0.615 |
| | NMI | 0.127 | 0.152 | 0.007 | 0.004 | 0.106 | 0.139 | 0.180 | 0.162 | 0.167 | 0.167 | 0.066 | 0.077 | 0.073 | 0.119 |
| | RL | - | 0.0179 | - | 0.0179 | - | 0.0179 | 0.0204 | 0.0189 | 0.0180 | 0.0174 | 0.0196 | 0.0187 | 0.0227 | 0.0190 |
| | RRL | - | 0.0 | - | 0.0 | - | 0.0 | 13.9 | 9.2 | 0.6 | 0.6 | **4.9**▼ | 4.5 | 10 | 9.8 |
| | ETP | 3.26 | 3.25 | 0.069 | 0.025 | 2.43 | 1.4 | 3.19 | 3.17 | 3.21 | 3.19 | 2.92 | 2.48 | 2.99 | 2.87 |
| | CS | 764-312 | 830-287 | 4969-1 | 4991-1 | 2586-82 | 3888-38 | 931-239 | 1003-263 | 844-191 | 804-258 | 2611-33 | 2076-21 | 912-45 | 1219-113 |
| CBird | SC | -0.019 | -0.021 | 0.037 | 0.094 | -0.026 | -0.062 | 0.311 | 0.251 | -0.032 | -0.037 | 0.171 | **0.448** | 0.446 | 0.312 |
| | NMI | 0.412 | 0.353 | 0.206 | 0.132 | 0.423 | 0.485 | 0.347 | 0.299 | 0.372 | 0.370 | 0.471 | 0.221 | 0.467 | 0.211 |
| | RL | - | 0.0055 | - | 0.0055 | - | 0.0055 | 0.0061 | 0.0040 | 0.0055 | 0.0036 | 0.0206 | 0.0060 | 0.0115 | 0.0039 |
| | RRL | - | 0.0 | - | 0.0 | - | 0.0 | 10 | 10 | 0.0 | 0.0 | 10 | 9.1 | **39**▼ | 8.3 |
| | ETP | 6.34 | 5.59 | 2.71 | 0.958 | 6.56 | 7.21 | 5.41 | 5.04 | 5.81 | 5.80 | 7.41 | 5.68 | 7.02 | 5.07 |
| | CS | 131-1 | 245-1 | 1722-1 | 2773-1 | 101-2 | 99-2 | 241-1 | 291-1 | 168-1 | 197-1 | 37-2 | 213-1 | 99-1 | 676-1 |
| R-10k | SC | -0.010 | - | 0.114 | - | -0.002 | - | - | - | - | - | 0.023 | - | **0.564** | - |
| | NMI | 0.398 | - | 0.012 | - | 0.383 | - | - | - | - | - | 0.152 | - | 0.367 | - |
| | RL | - | - | - | - | - | - | - | - | - | - | 0.0011 | - | 0.0011 | - |
| | RRL | - | - | - | - | - | - | - | - | - | - | **10** | - | **10** | - |
| | ETP | 5.13 | - | 0.072 | - | 5.10 | - | - | - | - | - | 5.51 | - | 4.77 | - |
| | CS | 916-20 | - | 11172-1 | - | 885-18 | - | - | - | - | - | 721-51 | - | 1046-1 | - |
| 20NG | SC | -0.021 | - | 0.114 | - | -0.008 | - | - | - | - | - | 0.101 | - | **0.197** | - |
| | NMI | 0.155 | - | 0.003 | - | 0.166 | - | - | - | - | - | 0.019 | - | 0.181 | - |
| | RL | - | - | - | - | - | - | - | - | - | - | 0.0009 | - | 0.0009 | - |
| | RRL | - | - | - | - | - | - | - | - | - | - | **10** | - | **10** | - |
| | ETP | 4.03 | - | 0.022 | - | 3.86 | - | - | - | - | - | 4.32 | - | 4.21 | - |
| | CS | 2217-107 | - | 18818-1 | - | 3428-26 | - | - | - | - | - | 1131-599 | - | 1812-199 | - |

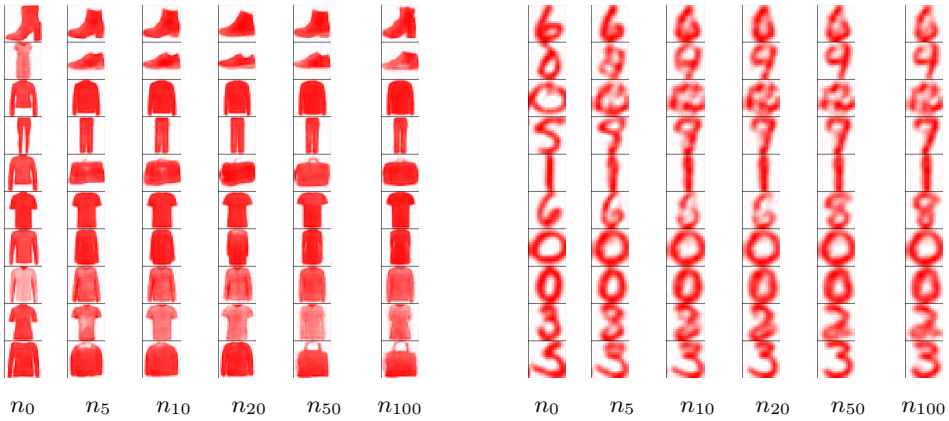
### C.3 How interpretable are the memories of DCAM?

We explore the representation of the learned prototypes in latent space for **DCAM** for Fashion-MNIST and USPS in Fig. 5. For Fashion-MNIST, the 60k images are partitioned into 10 clusters, and the evolution of cluster prototypes is visualized in Fig. 5a during the training process outlined in Algorithm 1 for **DCAM**. In each sub-figure we observe the evolution over epochs. At epoch 0, there are no distinct prototypes for clustering; instead, there are pairs of pullover (rows 3 and 5), shirts (rows 7 and 8), and t-shirts/tops (rows 6 and 9). However, discernible patterns emerge at epoch 10, refining further by epoch 20. By epoch 100, all ten prototypes represent distinct shapes, representing different cluster centroids.

Figure 6 provides a visualization of the 20 data points from the Fashion-MNIST dataset that are closest to each learned prototype. In the figure, the leftmost column represents the 10 learned prototypes stacked vertically, with each corresponding to one cluster in the dataset. The subsequent columns show the 20 data points that are nearest to each prototype or cluster center in terms of the Euclidean distance within the learned latent space. This visualization offers valuable insights into the nature of the learned prototypes and the clusters they represent. By examining the closest points, we can see how well the prototypes capture the underlying data distribution and whether they correspond to semantically meaningful clusters. For example, the learned prototypes often represent exemplar or central examples of specific fashion items (such as shirts, trousers, or shoes), while the nearest points reveal variations of these items that still fall within the same cluster. Such visualizations not only highlight the interpretability of **DCAM** 's energy-based clustering but also demonstrate the model's ability to learn compact and meaningful representations of the data. This approach provides an intuitive way to assess the quality of the learned clusters and their alignment with the dataset's inherent structure.

Table 10: **Metrics obtained by DCAM and baselines corresponding to the best RL (SC within 10% of the best SC of the method).** The best performance for each dataset is in **boldface**. (note abbreviations DCEC→DC, EDCWRN→EDC, Entropy→ETP, Cluster-size→CS, No-AE→NAE, Conv-AE→CAE, EDCWRN-AE→EAE, Resnet-AE→RAE). '-' denotes NA. x▼ indicates negative RRL which means the RL of the method is x% less than the pretrained AE loss.

| Data | Met | Kmeans | | Agglo | | C1AM | | DC | | DEKM | | EDC | DCAM | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | NAE | CAE | NAE | CAE | NAE | CAE | CAE | RAE | CAE | RAE | | CAE | EAE | RAE |
| FM | SC | 0.154 | 0.257 | 0.109 | 0.201 | 0.158 | 0.279 | 0.898 | 0.824 | 0.785 | 0.742 | 0.521 | 0.891 | 0.715 | **0.922** |
| | NMI | 0.511 | 0.643 | 0.534 | 0.624 | 0.521 | 0.622 | 0.561 | 0.623 | 0.571 | 0.633 | 0.493 | 0.472 | 0.522 | 0.401 |
| | RL | - | 0.0122 | - | 0.0122 | - | 0.0122 | 0.0109 | 0.0105 | 0.0514 | 0.0516 | 0.0211 | 0.0102 | 0.0131 | 0.0118 |
| | RRL | - | 0.0 | - | 0.0 | - | 0.0 | 9.8▼ | 26.5 | 321 | 522 | 143 | **16.4▼** | 54.0 | 42.2 |
| | ETP | 3.17 | 3.17 | 3.14 | 3.2 | 2.81 | 2.80 | 3.21 | 3.6 | 3.15 | 3.16 | 3.09 | 2.83 | 3.16 | 2.98 |
| | CS | 9617-2361 | 11145-2744 | 11830-1860 | 10298-2544 | 19032-1524 | 15679-2 | 11307-2766 | 9450-3132 | 12720-2478 | 11178-2658 | 13199-1391 | 17040-504 | 11886-2148 | 11830-1290 |
| C-10 | SC | 0.050 | 0.084 | 0.158 | 0.372 | 0.073 | 0.208 | 0.786 | 0.667 | 0.622 | 0.461 | 0.541 | **0.809** | 0.731 | 0.697 |
| | NMI | 0.078 | 0.122 | 0.0005 | 0.0004 | 0.073 | 0.015 | 0.099 | 0.094 | 0.092 | 0.119 | 0.111 | 0.079 | 0.060 | 0.082 |
| | RL | - | 0.0220 | - | 0.0220 | - | 0.0220 | 0.0217 | 0.0217 | 0.0616 | 0.0502 | 0.0291 | 0.0175 | 0.0252 | 0.0171 |
| | RRL | - | 0.0 | - | 0.0 | - | 0.0 | 0.9▼ | 20 | 180 | 179 | 74.3 | **20.4▼** | 50.9 | 5▼ |
| | ETP | 3.27 | 3.19 | 0.006 | 0.003 | 2.50 | 0.24 | 3.15 | 2.99 | 2.01 | 3.07 | 3.25 | 2.83 | 2.64 | 2.50 |
| | CS | 7105-2734 | 9779-2524 | 49979-1 | 49991-1 | 23544-582 | 48234-1 | 7145-4055 | 11025-1542 | 23420-26 | 14530-2505 | 8172-2562 | 17520-390 | 14890-120 | 17121-455 |
| C-100 | SC | 0.015 | -0.020 | 0.028 | 0.149 | 0.018 | 0.053 | 0.572 | 0.557 | 0.047 | 0.036 | 0.337 | 0.540 | 0.617 | **0.921** |
| | NMI | 0.161 | 0.183 | 0.036 | 0.004 | 0.153 | 0.156 | 0.158 | 0.119 | 0.162 | 0.221 | 0.186 | 0.112 | 0.201 | 0.094 |
| | RL | - | 0.0070 | - | 0.0070 | - | 0.0070 | 0.0083 | 0.0047 | 0.0679 | 0.0494 | 0.0128 | 0.0061 | 0.0092 | 0.0051 |
| | RRL | - | 0.0 | - | 0.0 | - | 0.0 | 18.6 | 17.5 | 870 | 1135 | 33.3 | **12.9▼** | 4.2▼ | 27.5 |
| | ETP | 6.53 | 6.48 | 0.940 | 0.052 | 6.51 | 4.38 | 5.8 | 4.06 | 6.18 | 6.11 | 6.51 | 4.02 | 5.83 | 3.22 |
| | CS | 1160-129 | 1395-23 | 38814-1 | 49834-1 | 1317-177 | 13950-11 | 2540-115 | 13736-12 | 1950-10 | 1980-10 | 996-156 | 11010-25 | 4350-10 | 22480-1 |
| USPS | SC | 0.143 | 0.195 | 0.124 | 0.158 | 0.144 | 0.194 | **0.929** | 0.909 | 0.882 | 0.843 | 0.491 | 0.914 | 0.911 | 0.914 |
| | NMI | 0.573 | 0.628 | 0.627 | 0.680 | 0.475 | 0.619 | 0.717 | 0.736 | 0.691 | 0.684 | 0.451 | 0.477 | 0.339 | 0.437 |
| | RL | - | 0.0019 | - | 0.0019 | - | 0.0019 | 0.0014 | 0.0029 | 0.0487 | 0.0558 | 0.0007 | 0.0025 | 0.0013 | 0.0025 |
| | RRL | - | 0.0 | - | 0.0 | - | 0.0 | **26.3▼** | 26.1 | 2463 | 2326 | 40 | 31.6 | 160 | 8.7 |
| | ETP | 3.27 | 3.23 | 3.26 | 3.27 | 3.10 | 3.16 | 3.27 | 3.27 | 3.24 | 3.25 | 3.29 | 3.11 | 2.55 | 2.99 |
| | CS | 284-121 | 359-89 | 333-121 | 328-104 | 420-53 | 375-64 | 283-106 | 283-127 | 334-87 | 312-97 | 294-156 | 463-35 | 947-27 | 513-49 |
| STL | SC | 0.039 | 0.079 | 0.158 | 0.270 | 0.051 | 0.108 | 0.766 | 0.812 | 0.745 | 0.804 | 0.431 | 0.919 | **0.923** | 0.865 |
| | NMI | 0.127 | 0.152 | 0.007 | 0.004 | 0.106 | 0.139 | 0.181 | 0.170 | 0.149 | 0.152 | 0.065 | 0.144 | 0.072 | 0.107 |
| | RL | - | 0.0179 | - | 0.0179 | - | 0.0179 | 0.0242 | 0.0310 | 0.0711 | 0.0578 | 0.0525 | 0.0354 | 0.0263 | 0.0255 |
| | RRL | - | 0.0 | - | 0.0 | - | 0.0 | 35.8 | 79.2 | 297 | 234 | 155 | 97.8 | **27.7** | 47.4 |
| | ETP | 3.26 | 3.25 | 0.069 | 0.025 | 2.43 | 1.4 | 3.23 | 3.26 | 1.15 | 3.22 | 2.90 | 2.48 | 2.98 | 2.86 |
| | CS | 764-312 | 830-287 | 4969-1 | 4991-1 | 2586-82 | 3888-38 | 725-229 | 741-299 | 4064-16 | 821-261 | 2641-23 | 2280-27 | 929-34 | 1466-69 |
| CBird | SC | -0.019 | -0.021 | 0.037 | 0.094 | -0.026 | -0.062 | 0.386 | 0.282 | 0.018 | 0.035 | 0.188 | 0.413 | **0.441** | 0.377 |
| | NMI | 0.412 | 0.353 | 0.206 | 0.132 | 0.423 | 0.485 | 0.333 | 0.297 | 0.316 | 0.273 | 0.484 | 0.222 | 0.466 | 0.209 |
| | RL | - | 0.0055 | - | 0.0055 | - | 0.0055 | 0.0229 | 0.0139 | 0.0625 | 0.0560 | 0.0377 | 0.0056 | 0.0104 | 0.0039 |
| | RRL | - | 0.0 | - | 0.0 | - | 0.0 | 316 | 286 | 1036 | 1455 | 102 | 1.8 | **44.4▼** | 8.3 |
| | ETP | 6.34 | 5.59 | 2.71 | 0.958 | 6.56 | 7.21 | 5.51 | 5.03 | 5.16 | 4.47 | 7.43 | 5.68 | 7.01 | 5.06 |
| | CS | 131-1 | 245-1 | 1722-1 | 2773-1 | 101-2 | 99-2 | 248-1 | 297-1 | 312-1 | 519-1 | 35-2 | 211-1 | 100-1 | 701-1 |
| R-10k | SC | -0.010 | - | 0.114 | - | -0.002 | - | - | - | - | - | 0.035 | - | **0.673** | - |
| | NMI | 0.398 | - | 0.012 | - | 0.383 | - | - | - | - | - | 0.147 | - | 0.378 | - |
| | RL | - | - | - | - | - | - | - | - | - | - | 0.0016 | - | 0.0022 | - |
| | RRL | - | - | - | - | - | - | - | - | - | - | **60** | - | 120 | - |
| | ETP | 5.13 | - | 0.072 | - | 5.10 | - | - | - | - | - | 5.55 | - | 4.79 | - |
| | CS | 916-20 | - | 11172-1 | - | 885-18 | - | - | - | - | - | 727-56 | - | 1026-1 | - |
| 20NG | SC | -0.021 | - | 0.114 | - | -0.008 | - | - | - | - | - | 0.099 | - | **0.287** | - |
| | NMI | 0.155 | - | 0.003 | - | 0.166 | - | - | - | - | - | 0.018 | - | 0.180 | - |
| | RL | - | - | - | - | - | - | - | - | - | - | 0.0006 | - | 0.0012 | - |
| | RRL | - | - | - | - | - | - | - | - | - | - | **25▼** | - | 50 | - |
| | ETP | 4.03 | - | 0.022 | - | 3.86 | - | - | - | - | - | 4.31 | - | 4.19 | - |
| | CS | 2217-107 | - | 18818-1 | - | 3428-26 | - | - | - | - | - | 1142-582 | - | 1809-197 | - |




(a) FM (b) USPS

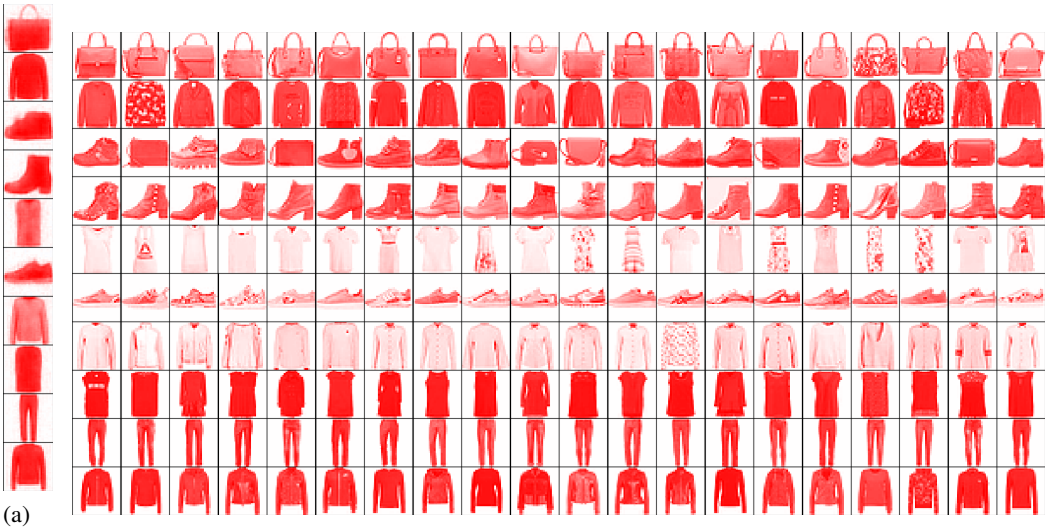
Figure 5: **Evolution of prototypes for Fashion-MNIST and USPS for DCAM.** We visualize the prototypes at the $n^{\text{th}}$ training epoch for $n = 0, 5, 10, 20, 50, 100$ (with $T = 10$).

### C.4 Effect of latent dimensionality

It is important to clarify that in **DCAM** the latent dimension $m$ is always set as the true number of classes per dataset, i.e., $m = k$. Indeed, most deep clustering schemes in the literature such as DCEC (Guo et al., 2017b), DEC (Xie et al., 2016), DEKM (Guo et al., 2021), and EDCWRN Oskouei et al. (2023) either follow this strategy or fix this to a specific number (e.g., 10), since latent representations are

Table 11: **Metrics obtained by DCAM and baselines corresponding to the best NMI.** The best performance for each dataset is in **boldface**. (note abbreviations DCEC→DC, EDCWRN→EDC, Entropy→ETP, Cluster-size→CS, No-AE→NAE, Conv-AE→CAE, EDCWRN-AE→EAE, Resnet-AE→RAE). '-' denotes NA. x▼ indicates negative RRL which means the RL of the method is x% less than the pretrained AE loss.

| Data | Met | Kmeans | | Agglo | | CLAM | | DC | | DEKM | | EDC | DCAM | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | NAE | CAE | NAE | CAE | NAE | CAE | CAE | RAE | CAE | RAE | | CAE | EAE | RAE |
| FM | SC | 0.154 | 0.251 | 0.109 | 0.201 | 0.140 | 0.262 | 0.861 | 0.716 | 0.819 | 0.784 | 0.430 | 0.817 | 0.619 | 0.825 |
| | NMI | 0.511 | 0.643 | 0.534 | 0.625 | 0.525 | 0.631 | 0.629 | **0.668** | 0.586 | 0.639 | 0.457 | 0.610 | 0.534 | 0.597 |
| | RL | - | 0.0122 | - | 0.0122 | - | 0.0122 | 0.0138 | 0.0139 | 0.0574 | 0.0596 | 0.0263 | 0.0406 | 0.0327 | 0.0387 |
| | RRL | - | 0.0 | - | 0.0 | - | 0.0 | 13.1 | 67.5 | 370 | 618 | 202 | 233 | 276 | 366 |
| | ETP | 3.17 | 3.17 | 3.14 | 3.20 | 3.13 | 2.98 | 3.22 | 3.20 | 3.07 | 3.16 | 3.00 | 3.16 | 3.22 | 3.18 |
| | CS | 9617-2361 | 11145-2744 | 11830-1860 | 10298-2544 | 14068-2435 | 15262-2100 | 10886-3030 | 9734-2847 | 12974-1191 | 11023-2652 | 17140-1578 | 11028-2658 | 10332-3054 | 10404-2610 |
| C-10 | SC | 0.050 | 0.072 | 0.014 | 0.020 | 0.064 | 0.101 | 0.118 | 0.653 | 0.276 | 0.262 | 0.541 | 0.713 | 0.632 | 0.420 |
| | NMI | 0.078 | 0.122 | 0.071 | 0.101 | 0.086 | 0.105 | 0.121 | 0.120 | 0.116 | 0.122 | 0.111 | **0.123** | 0.114 | 0.119 |
| | RL | - | 0.0220 | - | 0.0220 | - | 0.0220 | 0.0221 | 0.0245 | 0.0426 | 0.0362 | 0.0291 | 0.0403 | 0.0379 | 0.0326 |
| | RRL | - | 0.0 | - | 0.0 | - | 0.0 | 0.5 | 36.1 | 93.6 | 101 | 74.3 | 83.2 | 127 | 81.1 |
| | ETP | 3.27 | 3.19 | 3.17 | 3.02 | 3.23 | 2.21 | 3.07 | 3.21 | 3.19 | 3.11 | 3.25 | 3.18 | 2.98 | 3.28 |
| | CS | 7105-2734 | 9779-2524 | 10505-1650 | 11278-1764 | 9587-2925 | 26395-361 | 11022-3374 | 10235-1968 | 10275-2454 | 13746-2168 | 8172-2562 | 8595-2365 | 10721-289 | 6843-3144 |
| C-100 | SC | 0.015 | -0.014 | -0.018 | -0.043 | 0.018 | 0.001 | 0.048 | 0.002 | -0.011 | -0.028 | 0.308 | 0.354 | 0.200 | 0.130 |
| | NMI | 0.161 | 0.183 | 0.150 | 0.167 | 0.153 | 0.170 | 0.162 | 0.179 | 0.186 | 0.189 | 0.186 | 0.219 | 0.225 | **0.239** |
| | RL | - | 0.0070 | - | 0.0070 | - | 0.0070 | 0.0072 | 0.0049 | 0.0112 | 0.0074 | 0.0398 | 0.0257 | 0.0250 | 0.0226 |
| | RRL | - | 0.0 | - | 0.0 | - | 0.0 | 2.9 | 22.5 | 60 | 85 | 315 | 267 | 160 | 465 |
| | ETP | 6.53 | 6.48 | 6.45 | 6.30 | 6.51 | 6.27 | 6.41 | 6.41 | 5.23 | 6.45 | 5.51 | 6.33 | 6.33 | 6.36 |
| | CS | 1160-129 | 1395-23 | 1299-77 | 2308-17 | 1317-177 | 2535-39 | 1623-14 | 1380-21 | 2213-32 | 1440-60 | 996-156 | 1210-5 | 2105-15 | 1315-5 |
| USPS | SC | 0.143 | 0.195 | 0.124 | 0.159 | 0.142 | 0.180 | 0.920 | 0.896 | 0.946 | 0.465 | 0.43 | 0.865 | 0.660 | 0.857 |
| | NMI | 0.573 | 0.628 | 0.627 | 0.680 | 0.564 | 0.640 | **0.737** | 0.736 | 0.728 | 0.701 | 0.451 | 0.689 | 0.583 | 0.660 |
| | RL | - | 0.0019 | - | 0.0019 | - | 0.0019 | 0.0074 | 0.0039 | 0.0748 | 0.0374 | 0.0006 | 0.0451 | 0.0322 | 0.0409 |
| | RRL | - | 0.0 | - | 0.0 | - | 0.0 | 289 | 69.6 | 3836 | 1526 | 20 | 2274 | 6340 | 1678 |
| | ETP | 3.27 | 3.23 | 3.26 | 3.27 | 3.27 | 3.21 | 3.27 | 3.27 | 3.24 | 3.24 | 3.29 | 3.11 | 3.24 | 3.23 |
| | CS | 284-121 | 359-89 | 333-121 | 328-104 | 290-132 | 343-73 | 284-108 | 282-107 | 298-80 | 318-91 | 294-156 | 396-35 | 385-107 | 308-72 |
| STL | SC | 0.039 | 0.074 | 0.024 | 0.021 | 0.042 | 0.069 | 0.822 | 0.837 | 0.109 | 0.079 | 0.332 | 0.388 | 0.597 | 0.280 |
| | NMI | 0.127 | 0.152 | 0.121 | 0.138 | 0.130 | 0.169 | **0.188** | 0.165 | 0.170 | 0.166 | 0.103 | 0.149 | 0.151 | 0.159 |
| | RL | - | 0.0179 | - | 0.0179 | - | 0.0179 | 0.0328 | 0.0362 | 0.0315 | 0.0174 | 0.0433 | 0.0409 | 0.0454 | 0.0364 |
| | RRL | - | 0.0 | - | 0.0 | - | 0.0 | 83.2 | 109 | 76.0 | 0.6 | 110 | 128 | 120 | 110 |
| | ETP | 3.26 | 3.25 | 3.02 | 3.02 | 3.24 | 2.82 | 3.24 | 3.28 | 3.21 | 3.20 | 2.62 | 3.13 | 3.18 | 3.15 |
| | CS | 764-312 | 830-287 | 1379-205 | 1373-130 | 945-317 | 1212-2 | 849-232 | 671-326 | 807-250 | 876-264 | 2173-121 | 982-46 | 929-232 | 938-181 |
| CBird | SC | -0.019 | -0.021 | -0.018 | -0.064 | -0.026 | -0.062 | 0.248 | 0.152 | -0.041 | -0.038 | 0.188 | 0.135 | 0.068 | 0.167 |
| | NMI | 0.412 | 0.353 | 0.469 | 0.439 | 0.423 | 0.485 | 0.356 | 0.320 | 0.364 | 0.370 | 0.484 | 0.421 | **0.493** | 0.385 |
| | RL | - | 0.0055 | - | 0.0055 | - | 0.0055 | 0.0229 | 0.0152 | 0.0066 | 0.0036 | 0.0377 | 0.0255 | 0.0237 | 0.0249 |
| | RRL | - | 0.0 | - | 0.0 | - | 0.0 | 316 | 322 | 20 | 0.0 | 102 | 364 | 26.7 | 592 |
| | ETP | 6.34 | 5.59 | 6.97 | 6.58 | 6.56 | 7.21 | 5.84 | 5.12 | 5.71 | 5.80 | 7.43 | 6.48 | 7.39 | 6.05 |
| | CS | 131-1 | 245-1 | 93-1 | 232-1 | 101-2 | 99-2 | 167-1 | 570-1 | 177-1 | 197-1 | 35-2 | 143-1 | 58-2 | 180-1 |
| R-10k | SC | -0.010 | - | -0.012 | - | -0.007 | - | - | - | - | - | 0.013 | - | 0.647 | - |
| | NMI | 0.398 | - | 0.404 | - | 0.394 | - | - | - | - | - | 0.169 | - | **0.414** | - |
| | RL | - | - | - | - | - | - | - | - | - | - | 0.0014 | - | 0.0020 | - |
| | RRL | - | - | - | - | - | - | - | - | - | - | 40 | - | 100 | - |
| | ETP | 5.13 | - | 5.15 | - | 5.22 | - | - | - | - | - | 5.47 | - | 5.2 | - |
| | CS | 916-20 | - | 845-18 | - | 650-41 | - | - | - | - | - | 478-76 | - | 540-1 | - |
| 20NG | SC | -0.021 | - | -0.186 | - | -0.103 | - | - | - | - | - | 0.066 | - | 0.199 | - |
| | NMI | 0.155 | - | 0.167 | - | 0.176 | - | - | - | - | - | 0.018 | - | **0.229** | - |
| | RL | - | - | - | - | - | - | - | - | - | - | 0.0006 | - | 0.0012 | - |
| | RRL | - | - | - | - | - | - | - | - | - | - | 25▼ | - | 50 | - |
| | ETP | 4.03 | - | 3.64 | - | 3.77 | - | - | - | - | - | 4.31 | - | 3.87 | - |
| | CS | 2217-107 | - | 4024-52 | - | 4227-103 | - | - | - | - | - | 1142-582 | - | 3203-105 | - |



(a) M

(b) 20-closest points

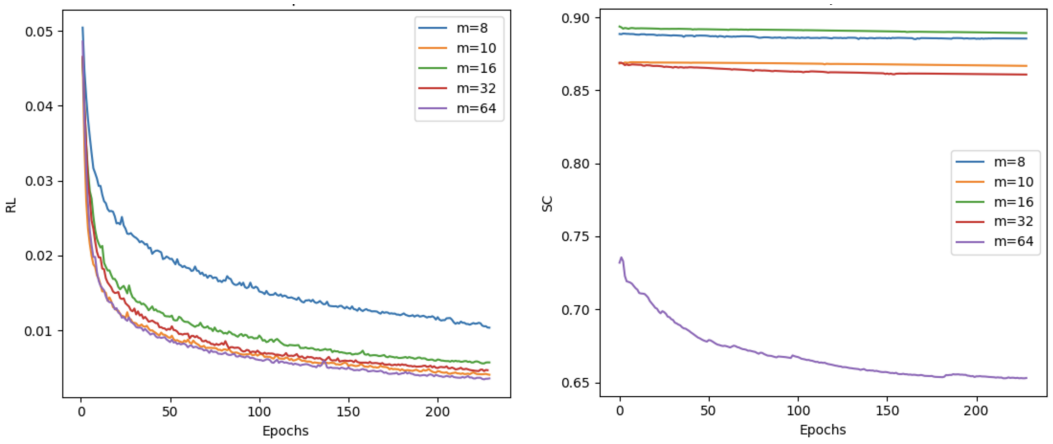
Figure 6: DCAM: **Final memories (left column) and the 20-closest points for each memory in F-MNIST.**

not only just good representations of the data points, they also represent the clusters. We thus always set this latent dimension as the number of clusters. By setting this exactly the same as the number of clusters, each latent dimension should ideally represent one specific cluster. If the latent dimensions

are larger than the number of clusters, some dimensions might not align with any specific cluster, or multiple dimensions could end up representing the same cluster. This can introduce redundancy and result in a less efficient representation. On the other hand, if the latent dimensions are smaller than the number of clusters, some clusters may not be adequately represented. This forces multiple clusters to share the same dimension, making it challenging for the model to distinguish between them accurately.

We can also consider a spectral argument (Von Luxburg, 2007) for setting $m = k$ . Given a set of $n$ points (in any representation) from $k$ ground-truth clusters, consider a graph (weighted or unweighted and undirected) with each point as a node, and edges between points belonging to the same cluster, and no inter-cluster edges. This graph would have $k$ connected components, and the Laplacian $L \in \mathbb{R}^{n \times n}$ of this graph will have $k$ zero eigenvalues (for example, see Von Luxburg [5, Proposition 2]). Now consider the first $k$ eigenvectors $u_1, \ldots, u_k \in \mathbb{R}^n$ forming the columns of the matrix $U \in \mathbb{R}^{n \times k}$. Then each row $z_i \in \mathbb{R}^k$ of $U$ can serve as a representation of the point $i$, and the points will be well-separated into $k$ clusters in this representation. This is the intuition that forms the basis of various spectral clustering algorithms.

The above implies the existence of a $k$-dimensional space where the points (coming from $k$ ground-truth clusters) are well-separated into $k$ clusters. Thus, a latent space of $k$ dimensions is necessary to obtain well-separated clusters. As Euclidean clustering becomes more challenging with increasing representation dimensionality (the representation in which the clustering is happening), the motivation is to keep the latent space dimension as low as possible as long as we have enough dimensions to separate the clusters. For this reason, the latent space dimensionality in most deep clustering methods usually matches the desired number of clusters. A higher latent dimensionality will definitely help with the reconstruction but can potentially hurt Euclidean clustering; a lower latent dimensionality would not be sufficient to obtain $k$ well-separated clusters. Fortunately, given the extremely expressive modern deep learning encoder and decoders, we are able to still get quite low reconstruction loss with a $m = k$ dimensional latent space.




(a) RL vs Training

(b) SC vs Training

Figure 7: Reconstruction loss (RL) and clustering quality (SC) for varying latent dimension ($m$) for USPS ($T$ is set to 10).

Despite the aforementioned caveats, Figure 7 illustrates the impact of varying the latent space dimensionality $m$ on the USPS dataset, which consists of 10 classes. The figure highlights the trade-offs between reconstruction loss (RL) and Silhouette Coefficient (SC) for different values of $m$. From the results, it is evident that setting $m = 10$ provides an optimal balance between RL and SC, achieving a strong trade-off compared to other values of $m$. This observation further supports our claim for matching the latent dimensionality with the number of clusters in the dataset. By doing so, **DCAM** effectively captures the underlying structure of the data while maintaining compact and well-separated clusters. This experiment underscores the importance of selecting an appropriate

latent dimensionality in clustering tasks and demonstrates how **DCAM** leverages this alignment to deliver meaningful and interpretable partitions.

### C.5 How **DCAM** loss relates to traditional deep clustering loss

Here, we show how the **DCAM** loss $\bar{\mathcal{L}}$ in Eq. (2) is related to the loss $\mathcal{L} = \mathcal{L}_r + \gamma\mathcal{L}_c$ in Eq. (1). If the encoder $\mathbf{e}$ and decoder $\mathbf{d}$ form a decent autoencoder (for example, if they are pretrained, as is common practice), then for a input $x \in S$, the single sample loss can be compared as follows:

$$\ell_r(x, \mathbf{e}, \mathbf{d}) \triangleq \|x - \mathbf{d}(\mathbf{e}(x))\|^2 \leq \|x - \mathbf{d}(A_{\boldsymbol{\rho}}^T(\mathbf{e}(x)))\|^2 \triangleq \bar{\ell}(x, \mathbf{e}, \mathbf{d}, \boldsymbol{\rho}), \tag{3}$$

since $A_{\boldsymbol{\rho}}^T(\mathbf{e}(x))$ will be some distortion of $\mathbf{e}(x)$, and thus its decoded version will generally be worse than the decoded version of $\mathbf{e}(x)$. Let us now assume that the decoder $\mathbf{d} : \mathbb{R}^m \to \mathbb{R}^d$ is $C_{\mathbf{d}}$-Lipschitz continuous. Then, considering the per-sample loss $\bar{\ell}$ in Eq. (2), and applying the triangle inequality and the AM–GM inequality, we can show that

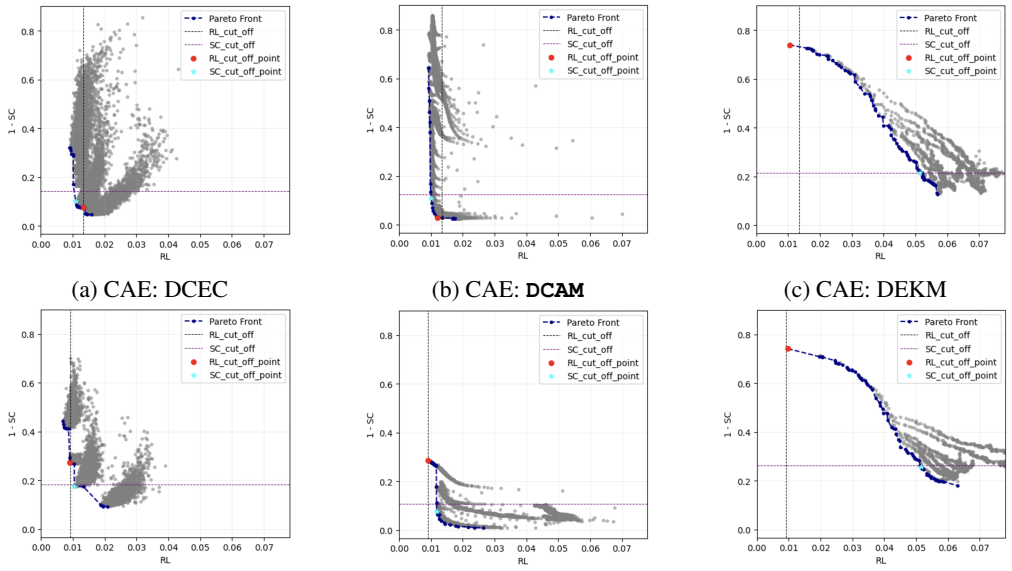
$$\begin{aligned}
\bar{\ell}(x, \mathbf{e}, \mathbf{d}, \boldsymbol{\rho}) &= \|x - \mathbf{d}(A_{\boldsymbol{\rho}}^T(\mathbf{e}(x)))\|^2 \\
&\leq 2\left(\|x - \mathbf{d}(\mathbf{e}(x))\|^2 + \|\mathbf{d}(\mathbf{e}(x)) - \mathbf{d}(A_{\boldsymbol{\rho}}^T(\mathbf{e}(x)))\|^2\right) \\
&\leq 2\left(\|x - \mathbf{d}(\mathbf{e}(x))\|^2 + C_{\mathbf{d}}^2\|\mathbf{e}(x) - A_{\boldsymbol{\rho}}^T(\mathbf{e}(x))\|^2\right) \\
&= 2\ell_r(x, \mathbf{e}, \mathbf{d}) + 2C_{\mathbf{d}}^2\ell_c(x, \mathbf{e}, \boldsymbol{\rho}),
\end{aligned} \tag{4}$$

where the last inequality uses the Lipschitz continuity, and the last equality comes from the definition of the clustering loss in the latent space with the AM dynamics operator. Summing the above inequalities in Eqs. (3) and (4) over $x \in S$ gives us $\mathcal{L}_r \leq \bar{\mathcal{L}} \leq \gamma_1\mathcal{L}_r + \gamma_2\mathcal{L}_c$, where the upperbound of $\bar{\mathcal{L}}$ is (a scaled version of) the standard deep clustering objective of the weighted combination of the reconstruction loss $\mathcal{L}_r$ and the clustering loss $\mathcal{L}_c$ in Eq. (1).

We would like to clarify that **DCAM** does not impose any specific constraints on the structure of the encoder and decoder (refer to Algorithm 1). In our discussion regarding Lipschitz continuity, our main goal is to highlight the relationship between the novel loss of **DCAM** and the loss of traditional deep clustering (Eq. (1) that consists of reconstruction and clustering losses). This comparison serves to underscore how the novel loss is related to the better intertwining of the different components of the deep clustering pipeline – the encoder, decoder, cluster centers. The novel **DCAM** loss provides significant improvements over Eq. (1) which uses the standard loss. Also note that if it is a decoder that we can differentiate through with auto-grad, the decoder is Lipschitz continuous. Additionally, there exists a more general notion called the modulus of continuity, which extends beyond Lipschitz continuity. We can substitute Lipschitz continuity with the modulus of continuity in our discussion, maintaining the same inequality but with potentially different constants.

### C.6 Additional details on hyperparameter selection

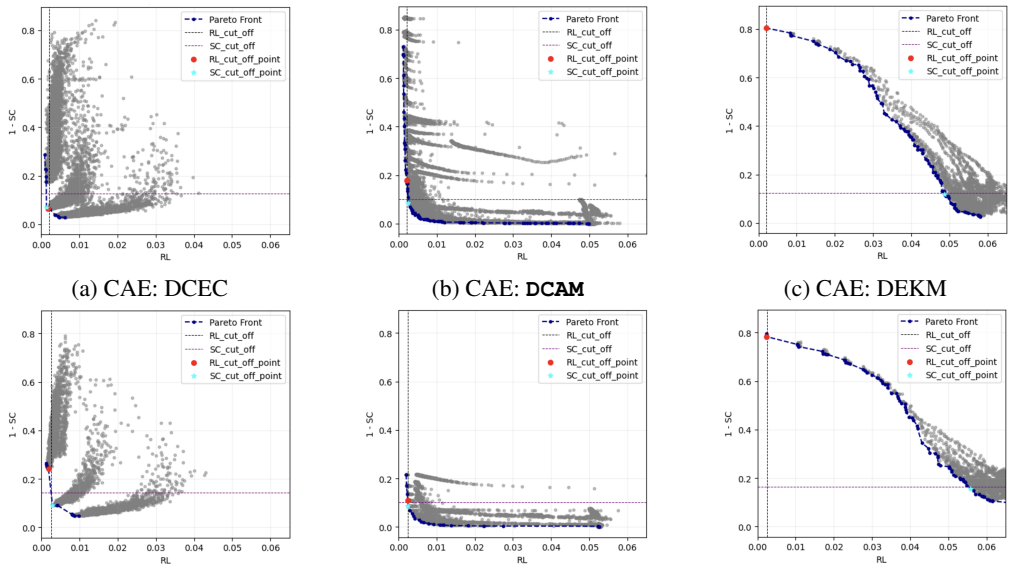
In Figs. 8 to 12, we plot the reconstruction loss (RL) and the silhouette coefficient (SC) for each hyperparameter configuration considered for **DCAM** and the baselines DCEC and DEKM for the different vision datasets (reported in Tables 1, 3, 4, 9, and 10). We also highlight the *Pareto front* for each of the dataset/method pairs, and the dotted vertical and horizontal lines denote the RL and (1-SC) values corresponding to the 10% margin from the best RL and (1-SC). Furthermore, the red and cyan highlighted points show the best hyperparameter configuration corresponding to the metric reported in Table 9 and 10. These results clearly highlight how we thoroughly optimize the hyperparameters, and how we select the final Pareto optimal performance values from the Pareto front to be consistent and fair across all methods. These results clearly show that **DCAM** offers the best clustering performance in terms of SC, as well as having low reconstruction loss. It also performs very well on the supervised NMI metric. In fact, for NMI, **DCAM** has the best value in 5 out of the 8 datasets (see Table 11).




(a) CAE: DCEC (b) CAE: **DCAM** (c) CAE: DEKM

(d) RAE: DCEC (e) RAE: **DCAM** (f) RAE: DEKM

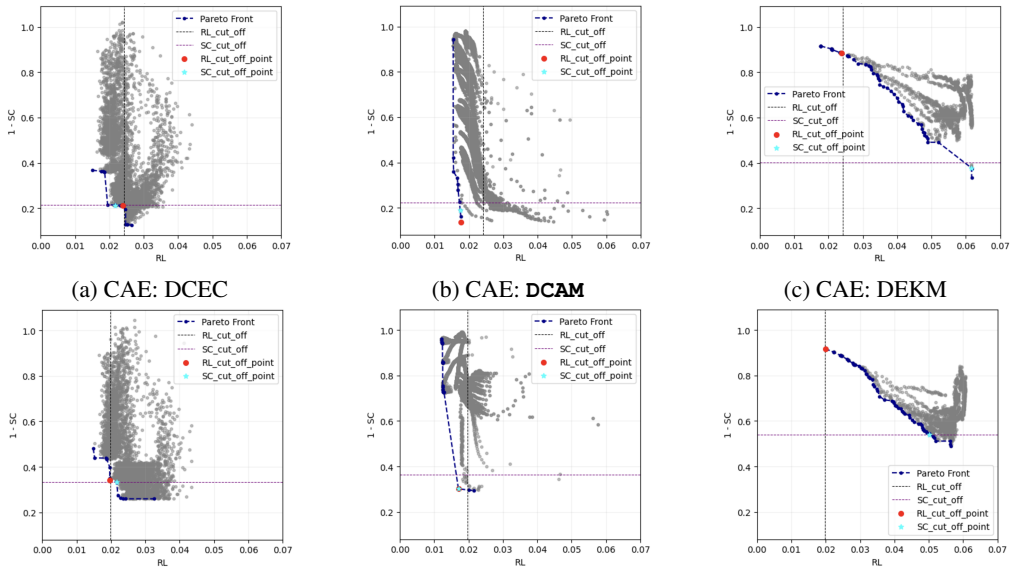
Figure 8: **FMNIST:** Reconstruction loss and clustering quality (1-SC) for all hyperparameter configurations for DCEC, **DCAM** and DEKM with CAE and RAE architectures. *Lower is better for both axes*, since we plot 1-SC on the $y$-axis.




(a) CAE: DCEC (b) CAE: **DCAM** (c) CAE: DEKM

(d) RAE: DCEC (e) RAE: **DCAM** (f) RAE: DEKM

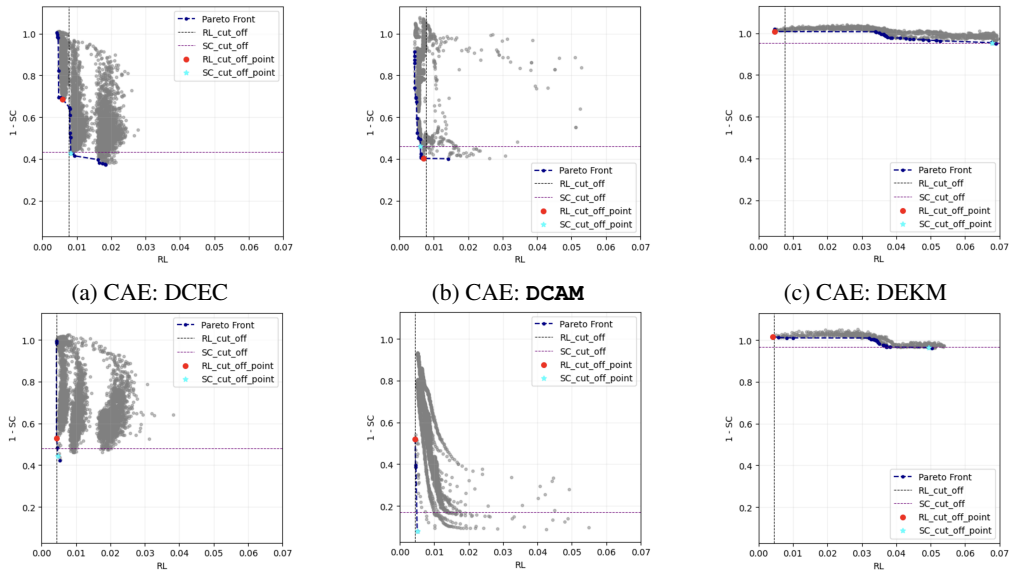
Figure 9: **USPS:** Reconstruction loss and clustering quality (1-SC) for all hyperparameter configurations for DCEC, **DCAM** and DEKM with CAE and RAE architectures. *Lower is better for both axes*.




(a) CAE: DCEC (b) CAE: **DCAM** (c) CAE: DEKM

(d) RAE: DCEC (e) RAE: **DCAM** (f) RAE: DEKM

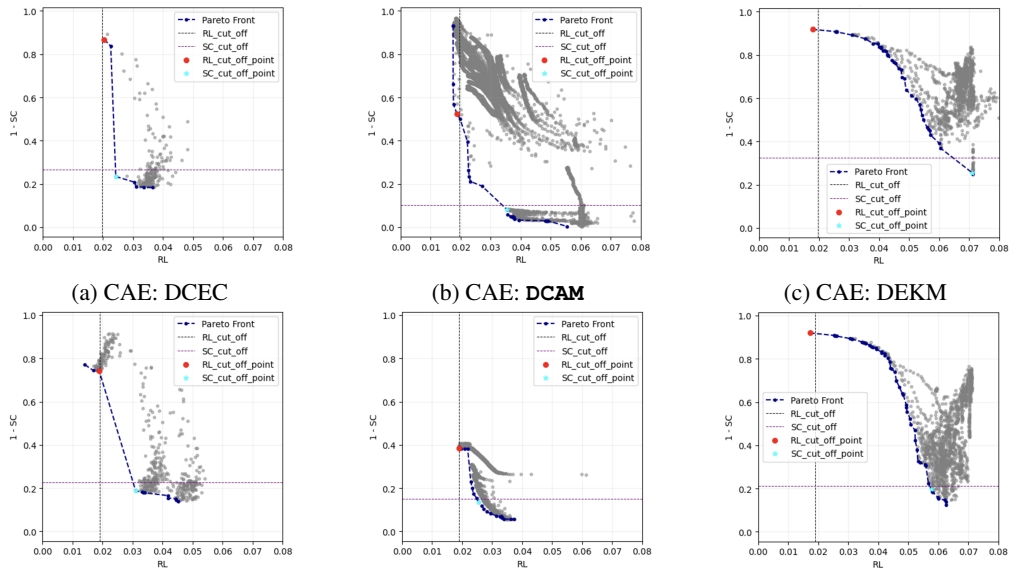
Figure 10: **CIFAR10:** Reconstruction loss and clustering quality (1-SC) for all hyperparameter configurations for DCEC, **DCAM** and DEKM with CAE and RAE architectures. *Lower is better for both axes*.




(a) CAE: DCEC (b) CAE: **DCAM** (c) CAE: DEKM

(d) RAE: DCEC (e) RAE: **DCAM** (f) RAE: DEKM

Figure 11: **CIFAR100:** Reconstruction loss and clustering quality (1-SC) for all hyperparameter configurations for DCEC, **DCAM** and DEKM with CAE and RAE architectures. *Lower is better for both axes*.




(a) CAE: DCEC (b) CAE: **DCAM** (c) CAE: DEKM

(d) RAE: DCEC (e) RAE: **DCAM** (f) RAE: DEKM

Figure 12: **STL10:** Reconstruction loss and clustering quality (1-SC) for all hyperparameter configurations for DCEC, **DCAM** and DEKM with CAE and RAE architectures. *Lower is better for both axes*.