Machine Learning
https://doi.org/10.1007/s10994-021-06015-5

Joint optimization of an autoencoder for clustering and embedding

Ahcène Boubekki[1] · **Michael Kampffmeyer**[1] · **Ulf Brefeld**[2] · **Robert Jenssen**[1]

Received: 22 November 2020 / Revised: 5 April 2021 / Accepted: 28 May 2021

Abstract
Deep embedded clustering has become a dominating approach to unsupervised categorization of objects with deep neural networks. The optimization of the most popular methods alternates between the training of a deep autoencoder and a k-means clustering of the autoencoder's embedding. The diachronic setting, however, prevents the former to benefit from valuable information acquired by the latter. In this paper, we present an alternative where the autoencoder and the clustering are learned simultaneously. This is achieved by providing novel theoretical insight, where we show that the objective function of a certain class of Gaussian mixture models (GMM's) can naturally be rephrased as the loss function of a one-hidden layer autoencoder thus inheriting the built-in clustering capabilities of the GMM. That simple neural network, referred to as the clustering module, can be integrated into a deep autoencoder resulting in a deep clustering model able to jointly learn a clustering and an embedding. Experiments confirm the equivalence between the clustering module and Gaussian mixture models. Further evaluations affirm the empirical relevance of our deep architecture as it outperforms related baselines on several data sets.

Keywords Clustering · Deep autoencoders · Embedding · k-means · Gaussian mixture models

Editors: Annalisa Appice, Sergio Escalera, Jose A. Gamez, Heike Trautmann.

✉ Ahcène Boubekki
ahcene.boubekki@uit.no

Michael Kampffmeyer
michael.c.kampffmeyer@uit.no

Ulf Brefeld
brefeld@leuphana.de

Robert Jenssen
robert.jenssen@uit.no

[1] Machine Learning Group, Department of Physics and Technology, UiT The Arctic University of Norway, Hansine Hansens veg 18, 9019 Tromsø, Norway

[2] Institute of Information Systems, Leuphana University of Lüneburg, Universitätsallee 1, 21335 Lüneburg, Germany

Published online: 21 June 2021

1 Introduction

Clustering is one of the oldest and most difficult problems in machine learning and a great deal of different clustering techniques have been proposed in the literature. Perhaps the most prominent clustering approach is the *k*-means algorithm (Lloyd, 1982). Its simplicity renders the algorithm particularly attractive to practitioners beyond the field of data mining and machine learning (Punj & Stewart, 1983; Gasch & Eisen, 2002; Hennig & Liao, 2013; Frandsen et al., 2015). A variety of related algorithms and extensions have been studied (Dunn, 1973; Krishna & Murty, 1999; Dhillon et al., 2004). The model behind the standard *k*-means algorithm is an isotropic Gaussian mixture model (Kulis & Jordan, 2012; Lücke & Forster, 2019) (GMM). As such, it can only produce linear partitions of the input space.

A dominating recent line of research aims to alleviate this issue by learning an embedding of the data with a deep autoencoder (DAE) and to have variants of Gaussian mixture models operate on the embedded instances (Xie et al., 2016; Yang et al., 2017; Guo et al., 2017; Fard et al., 2020; Miklautz et al., 2020). This can be referred to as *deep embedded clustering*. An important reason for the emergence of this modern line of research in clustering is that it combines elements (the deep autoencoder and the GMM clustering) that are theoretically well understood when considered separately. In practice, since the embedding map is nonlinear, the result is that the otherwise linear clustering translates to a nonlinear solution in input space. This approach resembles the rationale behind kernel methods (Guyon et al., 1993). The difference lies in either computing a kernel matrix before learning the clustering versus learning the feature map as the function approximated by the encoder. Another advantage of an autoencoder (AE) based approach is provided by the decoder of the network that allows to map the centroids back into (an approximated version of the) input space. Depending on the kernel function/matrix, this is not always possible with traditional approaches.

However, even with advantages of deep embedded clustering as described above, the fact that the embedding procedure and the clustering procedure are decoupled means that the former cannot benefit from the latter, and vice versa. In the most naive approach, the deep autoencoder is first trained, and then the features output by the autoencoder are clustered. In Xie et al. (2016), an improvement is made by alternatively shrinking the clusters in the embedded space around centroids for then to update the latter using *k*-means. The shrinking may however lead to an adaptation of the clustering to the embedding instead of the desired clustering-induced embedding. In general, these approaches cannot handle a random initialization which usually leads to noisy embeddings and, in turn, meaningless clusterings.

Despite the obvious need, few deep embedded clustering procedures have been proposed featuring an inherent capability of bringing out clustering structure in a joint and simultaneous procedure. This is clearly due to the lack of a theoretical foundation that could support such an objective. In this paper, we provide novel theoretical insight that enables us to simultaneously optimize both the embedding and the clustering by relying on a neural network interpretation of Gaussian mixture models. Our solution derives from the fact that during the EM, the objective function of a certain class of GMM can be rephrased as the loss function of an autoencoder, as we show as a key result. A network trained with that loss is thus guaranteed to learn a linear clustering. Our contributions are threefold. (1) We state and prove a theorem connecting Gaussian mixture models and autoencoders. (2) We define the clustering module (CM) as a one hidden layer AE with softmax activation

trained with the loss resulting from the previous point. (3) We integrate the CM into a deep autoencoder to form a novel class of deep clustering architectures, denoted AE-CM. Building on the theoretical insight, the experiments confirm the empirical equivalence of the clustering module with GMM. As for the AE-CM, the model outperforms on several datasets baselines extending Gaussian mixture models to nonlinear clustering with deep autoencoders.

The remainder of the paper is organized as follows. Section 2 reviews related works and baselines and Sect. 3 provides the theory underpinning our contributions. The clustering module and the AE-CM are introduced in Sect. 4. We report empirical evaluations in Sect. 5. Section 6 concludes the paper.

2 Related work

Recently, several deep learning models have been proposed to group data in an unsupervised manner (Tian et al., 2014; Springenberg 2016; Yang et al., 2016; Kampffmeyer et al., 2019; Ghasedi Dizaji et al., 2017; Haeusser et al., 2018; McConville et al., 2019; Mukherjee et al., 2019; Uğur et al., 2020; Van Gansbeke et al., 2020). Since our main contribution unravels the connection between Gaussian mixture models and autoencoders, models based on associative (Haeusser et al., 2018; Yang et al., 2019), spectral (Tian et al., 2014; Yang et al., 2019; Bianchi et al., 2020) or subspace (Zhang et al., 2019; Miklautz et al., 2020) clusterings are outside the scope of this paper. Furthermore, unlike (Yang et al., 2016; Chang et al., 2017; Kampffmeyer et al., 2019; Ghasedi Dizaji et al., 2017; Guo et al., 2017; Caron et al., 2018; Ji et al., 2019; Van Gansbeke et al., 2020), our model falls within the category of general purpose deep embedded clustering models which builds upon GMM (Xie et al., 2016; Guo et al., 2017; Yang et al., 2017; Fard et al., 2020). We mention that recent works related to specific topics such as perturbation robustness and non-redundant subspace clustering also utilize ideas that are to some degree related to the general concept of deep embedded clustering (Yang et al., 2020; Miklautz et al., 2020). Given the high generality of our method, we detail models that are also built without advanced mechanisms, but can still support them.

One of the first approaches in this latter category of dominating modern deep clustering algorithms is DEC (Xie et al., 2016), which consists of an embedding network coupled with an ad-hoc matrix representing the centroids. During training, the network is trained to shrink the data around their closest centroids in a way similar to t-SNE (Maaten & Hinton, 2008). However, instead of matching the distributions in the input and feature spaces, the target distribution is a mixture of Student-t models inferred from a k-means clustering of the embedding itself. The latter is updated every few iterations to keep track of evolution of the embedding and the centroids are stored in the ad-hoc matrix. In order to avoid sub-optimal solutions when starting from random initialization, the embedding network is pre-trained. Nevertheless, the difference between the two training phases can cause stability issues, such as collapsing centroids. IDEC (Guo et al., 2017) extends DEC and alleviates this issue to some degree by using the reconstruction loss of the DAE also during the training phase. Although IDEC is more robust than DEC, both are highly dependent on the initial embedding as the clustering phase quickly shrinks the data around the centroids. The Deep Clustering Network (DCN) (Yang et al., 2017) further strengthens the deep embedding clustering framework and is based on an architecture comparable to DEC and IDEC but includes hard clustering. The optimization scheme consists of three steps: (1) Optimize

the DAE to reconstruct and bring the data-points closer to their closest centroids (similar to k-means); (2) update assignments to the closest clusters; (3) update the centroids iteratively using the assignments. Although alternating optimization helps prevent instabilities during the optimization, DCN still requires initialization schemes in order to reach good solutions. Our proposed model instead, while being theoretically-driven, resolves the need for alternative optimization schemes the previous methods rely on. Since we learn a variant of GMM using a neural network, the optimizations of both the embedding and the clustering profit from each other. In practice, our model is thus less reliant on pre-training schemes.

To some degree our approach is similar the very recent method known as DKM (Fard et al., 2020). The model shares the same architecture as DEC, namely, a deep autoencoder and an ad-hoc matrix representing the centroids. However, here, the model optimizes the network and the centroids simultaneously. To achieve that the loss function includes a term similar to the loss of fuzzy c-means (Bezdek, 2013), i.e, k-means with soft-assignments. The cluster coefficients are computed from the distances to the centroids and normalized using a parameterized softmax. In order to convert soft assignments into hard ones, the parameter follows an annealing strategy. In our model, the assignment probability function is indirectly related to the distance to the closest cluster representative, as used in DKM. Our the loss function aims to minimize the distance between the data points and their reconstruction as convex combinations of the centroids. This means that the learning of the centroid and of the assignment probabilities regularize each other, mitigating the need of an annealing strategy.

There are also other recent and promising approaches to clustering that are leveraging deep neural networks. However, these operate in a very different manner compared to deep embedded clustering, which our novel theoretical insight sheds new light on and which enables our new proposed autoencoder optimization for joint clustering and embedding. We nevertheless briefly review some of these approaches and also compare to representatives of such methods in our experiments. In particular, a great deal of works have focus on the generative aspect of Gaussian mixture models and studied variations based on deep generative models such as variational autoencoders (VAE) (Kingma & Welling, 2013) and generative adversarial network (GAN) (Goodfellow et al., 2014).

Adversarial approaches for clustering have been proposed with examples being CatGAN (Springenberg, 2016) and adversarial autoencoders (AAE) (Makhzani et al., 2015). The former optimizes the mutual information and predicts a categorical distribution of classes in the data while maximizing at the same time the robustness against an adversarial generative model. Instead, AAE uses two adversarial networks to impose a Gaussian and a categorical distribution in the latent space. The recent ClusterGAN (Mukherjee et al., 2019) makes an original use of an autoencoder as the input is not a data point but a sample drawn from the product of a multinomial and a Gaussian distributions. The decoded sample is then fed to the discriminator and the encoder. This method is all robust to random network initialization.

Variational approaches that enable clustering include methods like Gaussian Mixture VAE (GMVAE) (Dilokthanakul et al., 2016) and Variational Deep Embedding (VaDE) (Jiang et al., 2016). Although the former explicitly refers to GMMs, both methods are similar. The difference is that the parameters of the Gaussian distributions in embedded space depend only on the cluster index in VaDE whereas GMVAE involves a second random variable. Both models require a pre-trained autoencoder. Variational information bottleneck with Gaussian mixture model (VIB-GMM) (Uğur et al., 2020) also assumes a mixture of Gaussian distributions on the embedding, however the model is trained accordingly to the deep variational information bottleneck (VIB) framework. The latter can be

seen as a information theory-driven generalization of GAN and VAE (Tishby et al., 2000; Alemi et al., 2016). VIB-GMM reveals to be a robust alternative to previous approaches as it is able to produce meaningful clusterings from a randomly initialized autoencoder.

3 Theoretical groundwork

Throughout the paper the set of positive integers is denoted by $\mathbb{N}^*$. The set of the d-dimensional stochastic vectors is written as $\mathbb{S}^d = \{\boldsymbol{x} \in \mathbb{R}^d_{\geq 0} \,:\, \sum_{i=1}^d x_i = 1\}$. When the context allows, the ranges of the indices are abbreviated using the upper-bound, e.g., a vector $\boldsymbol{x} \in \mathbb{R}^d$ decomposes as $\boldsymbol{x} = \langle x_j \rangle_{1 \leq j \leq d} = \langle x_j \rangle_d$. The zero vector of $\mathbb{R}^d$ is written as $\mathbf{0}_d$. The notation extends to any number. The identity matrix of $\mathbb{R}^{d \times d}$ is denoted by $\mathbf{I}_d$.

3.1 From GMMs to Autoencoders

We aim to fit an isotropic Gaussian mixture model with $K \in \mathbb{N}^*$ components and a Dirichlet prior on the average cluster responsibilities on a dataset $\mathcal{X} = \{\boldsymbol{x}_i\}_N \subset \mathbb{R}^d$. The *expected value of the complete-data log-likelihood* function of the model, also called the $\mathcal{Q}$ -function (Bishop, 2006) is:

$$\mathcal{Q}(\boldsymbol{\Gamma}, \boldsymbol{\Phi}, \boldsymbol{\mu}) = \sum_{i=1}^{N} \sum_{k=1}^{K} \gamma_{ik} \big(\log \phi_k - ||\boldsymbol{x}_i - \boldsymbol{\mu}_k||^2 \big) + \sum_{k=1}^{K} (\alpha_k - 1) \log \tilde{\boldsymbol{\gamma}}_k. \tag{1}$$

In this expression, $z_i \in \mathbb{N}$ is the cluster assignment of the data point $\boldsymbol{x}_i$, $\gamma_{ik} := p(z_i = k | \boldsymbol{x}_i)$ is the posterior probability of $z_i = k$, also called the responsibility of cluster k on a datapoint $\boldsymbol{x}_i$, $\phi_k := p(z_i = k)$ is the prior probability or mixture weight of cluster k and $\boldsymbol{\mu}_k \in \mathbb{R}^d$ is the centroid of cluster k. The average responsibilities of cluster k is $\tilde{\gamma}_k = \frac{1}{N} \sum_{i=1}^{N} \gamma_{ik}$ and $\tilde{\boldsymbol{\gamma}} \in \mathbb{S}^K$. The concentration hyperparameter of the Dirichlet prior is $\langle \alpha_k \rangle_K \in \mathbb{R}^K \backslash \{\mathbf{0}_K\}$. The co-variance matrices do not appear in this expression as they are all constant and equal to $\frac{1}{2}\mathbf{I}_d$ due to the isotropic assumption. We summarize the parameters into matrices $\boldsymbol{\Gamma} \in \mathbb{R}^{N \times K}$, $\boldsymbol{\Phi} \in \mathbb{S}^K$ and $\boldsymbol{\mu} \in \mathbb{R}^{K \times d}$. Note that the cluster responsibility vector of $\boldsymbol{x}_i$ is stochastic, i.e., $\boldsymbol{\gamma}_i = \langle \gamma_{ik} \rangle_K \in \mathbb{S}^K$.

Although, the isotropic co-variances allow for great simplifications, they restrict the model to spherical Gaussian distributions. We later alleviate this constraint by introducing a deep autoencoder. The Dirichlet prior involves an extra parameter but also smoothens the optimization. Note that these two assumptions make the model a relaxation of the one underlying k-means (Kulis & Jordan, 2012; Lücke & Forster, 2019).

Theorem 1 *The maximization of the $\mathcal{Q}$-function in Eq. (1) with respect to $\boldsymbol{\Phi}$ yields a term that can be interpreted as the reconstruction loss of an autoencoder.*

Proof During the EM-algorithm, the maximization with respect to $\boldsymbol{\Phi}$ updates the latter as the average cluster responsibility vector:

$$\phi_k = \frac{1}{N} \sum_{i=1} \gamma_{ik} = \tilde{\gamma}_k.$$

Using this result, the $\mathcal{Q}$-function can be rephrased as

$$\mathcal{Q}(\boldsymbol{\Gamma}, \boldsymbol{\mu}) = \sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ik} \log \tilde{\boldsymbol{\gamma}}_k - \sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ik} ||\boldsymbol{x}_i - \boldsymbol{\mu}_k||^2 + \sum_{k=1}^{K} (\alpha_k - 1) \log \tilde{\boldsymbol{\gamma}}_k. \tag{2}$$

The first term corresponds to the entropy of $\langle \tilde{\gamma}_k \rangle_K$ which, given the Dirichlet prior, is constant and can thus be omitted. We expand now $\gamma_{ik} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2$ by adding and subtracting the norm of $\bar{\boldsymbol{x}}_i = \sum_{k=1}^{K} \gamma_{ik} \boldsymbol{\mu}_k$: For any given $i \in [1 \,..\, N]$,

$$\begin{aligned}
\sum_{k=1}^{K} \gamma_{ik} ||\boldsymbol{x}_i - \boldsymbol{\mu}_k||^2 &= \sum_{k=1}^{K} \gamma_{ik} ||\boldsymbol{x}_i||^2 - 2\boldsymbol{x}_i^\top \Big(\sum_{k=1}^{K} \gamma_{ik} \boldsymbol{\mu}_k \Big) + \sum_{k=1}^{K} \gamma_{ik} ||\boldsymbol{\mu}_k||^2 \\
&\quad + ||\bar{\boldsymbol{x}}_i||^2 - ||\bar{\boldsymbol{x}}_i||^2 \\
&= ||\boldsymbol{x}_i||^2 - 2\boldsymbol{x}_i^\top \bar{\boldsymbol{x}}_i + ||\bar{\boldsymbol{x}}_i||^2 \\
&\quad + \sum_{k=1}^{K} \gamma_{ik} ||\boldsymbol{\mu}_k||^2 - \sum_{k=1}^{K}\sum_{l=1}^{K} \gamma_{ik}\gamma_{il} \boldsymbol{\mu}_k^T \boldsymbol{\mu}_k \\
&= ||\boldsymbol{x}_i - \bar{\boldsymbol{x}}_i||^2 + \sum_{k=1}^{K} \gamma_{ik}(1-\gamma_{ik}) ||\boldsymbol{\mu}_k||^2 - \sum_{k=1}^{K} \sum_{\substack{l=1 \\ l \neq k}}^{K} \gamma_{ik}\gamma_{il} \boldsymbol{\mu}_k^T \boldsymbol{\mu}_l.
\end{aligned}$$

Note that this simplification grounds on the isotropic assumption. It is not obvious how to reach a similar result without it. The $\mathcal{Q}$-function becomes:

$$\begin{aligned}
\mathcal{Q}(\boldsymbol{\Gamma}, \boldsymbol{\mu}) = &\underbrace{- \sum_{i=1}^{N} ||\boldsymbol{x}_i - \bar{\boldsymbol{x}}_i||^2}_{=:E_1} \underbrace{- \sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ik}(1-\gamma_{ik}) ||\boldsymbol{\mu}_k||^2}_{=:E_2} \\
&\underbrace{+ \sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{\substack{l=1 \\ l \neq k}}^{K} \gamma_{ik}\gamma_{il} \boldsymbol{\mu}_k^T \boldsymbol{\mu}_l}_{=:E_3} \underbrace{- \sum_{k=1}^{K} (1-\alpha_k) \log \tilde{\gamma}_k}_{=:E_4}.
\end{aligned} \tag{3}$$

The variable $\bar{\boldsymbol{x}}_i$ is actually a function of $\boldsymbol{x}$ that factorizes into two functions $\mathcal{F}$ and $\mathcal{G}$. The former computes $\boldsymbol{\gamma}_i$ which, as a posterior probability, is a function of $\boldsymbol{x}_i$. The latter is the dot-product with $\boldsymbol{\mu}$.

$$\begin{aligned}
\mathcal{F} : \begin{array}{lll} \mathbb{R}^d & \to & \mathbb{S}^K \\ \boldsymbol{x} & \mapsto & \boldsymbol{\mathcal{F}}(\boldsymbol{x};\boldsymbol{\eta}) = \langle p(z=k|\boldsymbol{x}) \rangle_K = \boldsymbol{\gamma} \end{array} \\
\mathcal{G} : \begin{array}{lll} \mathbb{S}^K & \to & \mathbb{R}^d \\ \boldsymbol{\gamma} & \mapsto & \boldsymbol{\mathcal{G}}(\boldsymbol{\gamma};\boldsymbol{\mu}) = \sum_{k=1}^{K} \gamma_k \boldsymbol{\mu}_k = \bar{\boldsymbol{x}} \end{array}
\end{aligned} \tag{4}$$

The first term of Eq. (3) can thus be interpreted as the reconstruction term characteristic of a loss of an autoencoder, consisting of the encoder and decoder functions, which are $\mathcal{F}$ and $\mathcal{G}$, respectively. □

3.2 Analysis of the terms

The four terms of the expansion of the $\mathcal{Q}$-function as Eq. (3) gives new insights into the training of GMMs.

E_1: Reconstruction The term E_1 suggests an autoencoder structure to optimize $\mathcal{Q}$. The decoder, $\mathcal{G}$, is a linear function. However, without loss of generality, it can be treated in practice as an affine function, i.e. a single layer network with bias. Regarding the encoder $\mathcal{F}$, the architecture can be anything as long as it has a stochastic output.

E_2: Sparsity and regularization The second term E_2 is related to the Gini impurity index (Breiman et al., 1984) applied to $\boldsymbol{\gamma}_i$:

$$\sum_{k=1}^{K} \gamma_{ik}(1-\gamma_{ik})\|\boldsymbol{\mu}_k\|^2 \leq \sum_{k=1}^{K} \gamma_{ik}(1-\gamma_{ik})\|\boldsymbol{\mu}\|_F^2 = \mathbf{Gini}(\boldsymbol{\gamma}_i)\|\boldsymbol{\mu}\|_F^2,$$

where $\|.\|_F$ is the Frobenius norm. The Gini index is standard in decision tree theory to select branching features and is an equivalent of the entropy. It is nonnegative and null if, and only if, $\boldsymbol{\gamma}_i$ is a one-hot vector. Hence, minimizing this term favors sparse $\boldsymbol{\gamma}_i$ resulting in clearer assignments.

The terms $\|\boldsymbol{\mu}_k\|^2$ play a role similar to an ℓ_2-regularization: they prevent the centroids from diverging away from the data-points. However, they may also favor the trivial solution where all the centroids are merged into zero.

E_3: Sparsity and cluster merging To study the behavior of E_3 during the optimization, let us consider a simple example with one observation and two clusters, i.e, $\boldsymbol{\Gamma} \equiv \boldsymbol{\gamma}_1 = (\gamma, 1-\gamma)$. If the observation is unambiguously assigned to one cluster, $\boldsymbol{\gamma}_1$ is a one-hot vector and E_3 is null. If it is not the case, the difference between E_2 and E_3 factorizes as follows:

$$E_2 - E_3 = \gamma(1-\gamma)\big(\|\boldsymbol{\mu}_1\|^2 + \|\boldsymbol{\mu}_2\|^2 - \boldsymbol{\mu}_1^T\boldsymbol{\mu}_2\big) = \gamma(1-\gamma)\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|^2. \tag{5}$$

The optimization will thus either push $\boldsymbol{\gamma}_1$ toward a more sparse vector, or merge the two centroids. Appendix 3 presents an analysis of the role of this term.

E_4: Balancing The Dirichlet prior steers the distribution of the cluster assignments. If none of the α_k is null, the prior will push the optimization to use all the clusters, moderating thus the penchant of E_2 for the trivial clustering (Yang et al., 2017; Guo et al., 2017).

Note that if $\boldsymbol{\alpha} = \left(1 + \frac{1}{K}\right)\mathbf{1}_K$, E_4 is, up to a constant, equal to the Kullback–Leibler (KL) divergence between a multinomial distribution with parameter $\tilde{\boldsymbol{\gamma}}$ and the uniform multinomial distribution:

$$\sum_{k=1}^{K}(1-\alpha_k)\log\tilde{\boldsymbol{\gamma}}_k = \sum_{k=1}^{K}(1-(1+\frac{1}{K}))\log\tilde{\boldsymbol{\gamma}}_k = D_{KL}\Big(\frac{1}{K}\mathbf{1}_K\Big\|\tilde{\boldsymbol{\gamma}}\Big).$$

4 Clustering and embedding with Autoencoders

Theorem 1 says that an autoencoder could be involved during the EM optimization of an isotropic GMM. We go one step further and by-pass the EM to directly optimize the $\mathcal{Q}$ -function in Eq. (3)) using an autoencoder.

4.1 The clustering module

We define the Clustering Module (CM) as the one-hidden layer autoencoder with encoding and decoding functions $\mathcal{F}$ and $\mathcal{G}$ such as:

$$\begin{aligned} \mathcal{F}(\boldsymbol{X}) &= \text{softmax}(\boldsymbol{X}\boldsymbol{W}_{\text{enc}} + \boldsymbol{B}_{\text{enc}}) = \boldsymbol{\Gamma} \\ \mathcal{G}(\boldsymbol{\Gamma}) &= \boldsymbol{\Gamma}\boldsymbol{W}_{\text{dec}} + \boldsymbol{B}_{\text{dec}} = \bar{\boldsymbol{X}}, \end{aligned} \tag{6}$$

where $\boldsymbol{X} \in \mathbb{R}^{N\times d} \sim \mathcal{X}$, code representation/cluster responsibilities $\boldsymbol{\Gamma} = \langle \gamma_{ik} \rangle_{N\times K} \in \mathbb{R}^{N\times K}$ s.t. $\boldsymbol{\gamma}_i \in \mathbb{S}^K$, and the reconstruction $\bar{\boldsymbol{X}} \in \mathbb{R}^{N\times d}$. The weight and bias parameters of the encoder are $\boldsymbol{W}_{\text{enc}} \in \mathbb{R}^{d\times K}$ and $\boldsymbol{B}_{\text{enc}} \in \mathbb{R}^K$, respectively, and analogously for the decoder $\boldsymbol{W}_{\text{dec}} \in \mathbb{R}^{K\times d}$ and $\boldsymbol{B}_{\text{dec}} \in \mathbb{R}^d$. The softmax enforces the row-stochasticity of the code, ie., $\boldsymbol{\Gamma}$. The associated loss function is the negative of Eq. (3):

$$\begin{aligned} \mathcal{L}_{\text{CM}}(\mathcal{X};\Theta) = &\sum_{i=1}^{N} ||\boldsymbol{x}_i - \bar{\boldsymbol{x}}_i||^2 + \sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ik}(1-\gamma_{ik})||\boldsymbol{\mu}_k||^2 \\ &- \sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{\substack{l=1 \\ l\neq k}}^{K} \gamma_{ik}\gamma_{il}\boldsymbol{\mu}_k^T\boldsymbol{\mu}_l + \sum_{k=1}^{K}(1-\alpha_k)\log\tilde{\gamma}_k. \end{aligned} \tag{7}$$

with $\Theta = \left(\boldsymbol{W}_{\text{enc}}, \boldsymbol{B}_{\text{enc}}, \boldsymbol{W}_{\text{dec}}, \boldsymbol{B}_{\text{dec}}\right)$. The centroids of the underlying GMM correspond to the images of $\mathcal{G}$ of the canonical basis of $\mathbb{R}^K$.

Initialization

The CM can be initialized using k-means or any initialization scheme thereof such as *k-means++* (Arthur & Vassilvitskii, 2007). In such a case, the column-vectors of $\boldsymbol{W}_{\text{dec}}$ are set equal to the desired centroids. The pseudo-inverse of this matrix becomes the encoder's weights, $\boldsymbol{W}_{\text{enc}}$, and both bias vectors are set to null.

Averaging epoch

In practice, the CM will be optimized by mini-batch learning with stochastic gradient descent. In such a procedure, the optimizer updates the positions of the centroids given the current batch. A small batch-size relative to the size of $\mathcal{X}$ may cause dispersion of the intermediate centroids. Hence, choosing the final centroids based on the last iteration may be sub-optimal.

We illustrate this phenomenon in Fig. 1. The data consists of $N = 2000$ points in $\mathbb{R}^2$ drawn from a mixture of five bi-variate Gaussians ($K = 5$) (gray dots). The data is standardized before processing. A CM is trained in mini-batches of size 20 over 50 epochs using stochastic gradient descent. The concentration is set to $\boldsymbol{\alpha} = \mathbf{5}_K$. The dispersion of the centroids after each iteration of the last epoch (crosses) is significant. On the other hand, their average positions (squares) provide a good approximation of the true centers (circles). Therefore, we include one extra epoch to any implementation of the CM to compute the average position of the individual centroids over the last iterations.

4.2 Embedding with feature maps

The theory behind GMMs limits the CM to a linear decoder, thus enabling merely a linear partition of the input space. In addition, the isotropy assumption, specific to the CM,

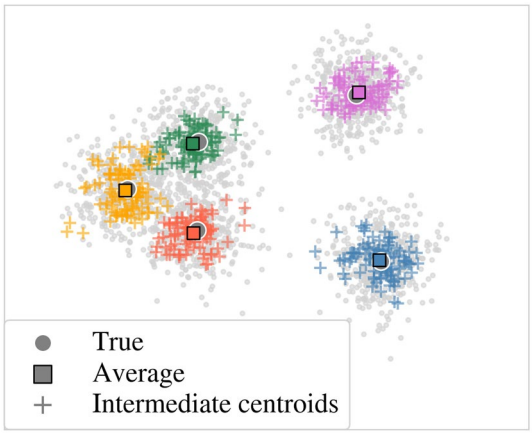

Fig. 1 The intermediate centroids of the last epoch are spread, whereas their averages almost match the true centroids

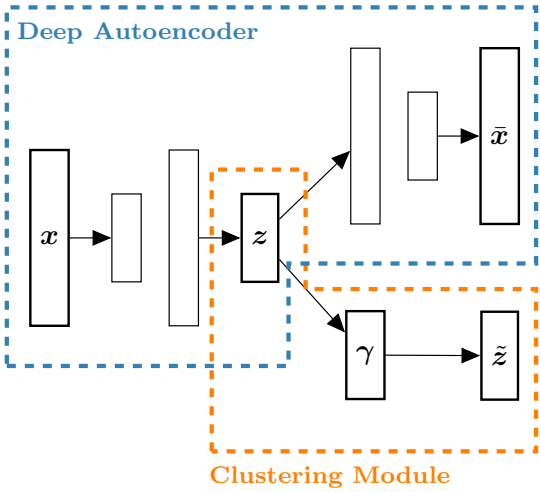

Fig. 2 Schematic representation of the AE-CM. Combining a clustering module and a deep autoencoder allows to jointly learn a clustering and an embedding

bars clusters to spread differently. We alleviate both *limitations* using a similar approach to that of kernel methods (Guyon et al., 1993): we non-linearly project the input into a feature space where it will be clustered. However, we do not learn the kernel matrices, providing an implicit feature map. Instead, we learn explicitly the feature maps using a deep autoencoder (DAE).

The idea is to optimize the CM and the DAE simultaneously, in order to let the latter find distortions of the input space along the way that guides the CM toward a better optimum. Using a deep autoencoder architecture prevents the optimization to produce degenerate feature maps Guo et al. (2017). It also preserves the generative nature of the model: points in the input space can be generated from a combination of centroids in the feature space. We refer to this model as the AE-CM.

The model consists of a clustering module nested into a deep autoencoder The architecture is illustrated in Fig. 2. The first part of the DAE encodes an input $\boldsymbol{x} \in \mathbb{R}^d$ into a vector $\boldsymbol{z} \in \mathbb{R}^p$. Note, CM now works on code representation $\boldsymbol{z}$ and not directly on the

input $\boldsymbol{x}$. The code $\boldsymbol{z}$ is fed to the CM and to the decoder of the DAE, yielding two outputs: $\tilde{\boldsymbol{z}}$, the CM's reconstruction of $\boldsymbol{z}$, and $\bar{\boldsymbol{x}}$, the DAE's reconstruction of $\boldsymbol{x}$.

4.2.1 Adapting the loss function to a deep architecture

Empirical evaluation showed that current gradient descent optimizers (e.g., Adam, Kingma & Ba, 2015) often return sub-optimal solutions when the reconstruction of the deep autoencoder is simply added to the loss of the CM. To help the optimization to find better optima, we add the assumption that the centroids are orthonormal:

$$\forall k, l,\ \boldsymbol{\mu}_k^T \boldsymbol{\mu}_l = \delta_{kl} = \begin{cases} 1 \text{ if } k = l, \\ 0 \text{ otherwise.} \end{cases} \tag{8}$$

Although the previous formula involves only $\boldsymbol{\mu}$ which is learned by the nested clustering module, it affects the surrounding DAE. Indeed, the constraint encourages it to produce an embedding where the centroids can simultaneously be orthonormal and minimize CM's loss. As a consequence, E_2 simplifies and E_3 becomes null:

$$\begin{aligned} E_2 &= \sum_{i=1}^{N} \sum_{k=1}^{K} \gamma_{ik}(1-\gamma_{ik}) ||\boldsymbol{\mu}_k||^2 = \sum_{i=1}^{N} \sum_{k=1}^{K} \gamma_{ik}(1-\gamma_{ik}) \\ E_3 &= \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{\substack{l=1 \\ l \neq k}}^{K} \gamma_{ik}\gamma_{il} \boldsymbol{\mu}_k^T \boldsymbol{\mu}_l = 0 \end{aligned} \tag{9}$$

Note that the constraint Eq. (8) is satisfied for the "ideal" clustering, in which case clusters will be mapped to the corners of a simplex in the embedding space, as discussed recently in Kampffmeyer et al. (2019). In this perspective, our inclusion of this constraint helps guide the clustering towards the ideal clustering, and at the same time simplifies the loss function.

We employ Lagrange multipliers to integrate the orthonormality constraint. That way the final loss can be stated as follows:

$$\begin{aligned} \mathcal{L}_{\text{AE-CM}}(\mathcal{X};\Theta) = &\beta \sum_{i=1}^{N} ||\boldsymbol{x}_i - \bar{\boldsymbol{x}}_i||^2 (\text{Reconstruction DAE}) \\ &+ \sum_{i=1}^{N} ||\boldsymbol{z}_i - \tilde{\boldsymbol{z}}_i||^2 (\text{Reconstruction CM}) \\ &+ \sum_{i=1}^{N} \sum_{k=1}^{K} \gamma_{ik}(1-\gamma_{ik}) (\text{Sparsity}) \\ &+ \sum_{k=1}^{K} (1-\alpha_k) \log(\tilde{\boldsymbol{\gamma}}_k) (\text{Dirichlet Prior}) \\ &+ \lambda ||\boldsymbol{\mu}^T \boldsymbol{\mu} - \mathbf{I}_K||_1, (\text{Orthonormality}) \end{aligned} \tag{10}$$

where $\lambda > 0$ is the Lagrange multiplier and $\beta > 0$ weights the DAE's reconstruction loss. We choose the ℓ_1-norm to enforce orthonormality, however other norms can be used.

Note that if the dimension of the embedding p is larger than the number of centroids K, the embedding can always be transformed to satisfy the orthonormality of the centroids. On the other hand, if $K > p$, the assumption becomes restrictive also in term of possible clusterings. Nevertheless, its importance can be reduced with a small λ. This assumption also helps to avoid the centroids to collapse as their norm is required to be 1. An analysis of the Lagrange multiplier is provided in Appendix 1.

The loss function of the AE-CM thus depends on four hyper-parameters: the weight $\beta > 0$, the concentration $\boldsymbol{\alpha} \in \mathbb{S}^K$, the Lagrange multiplier $\lambda > 0$, and the size of the batches $B \in \mathbb{N}^*$.

4.2.2 Implementation details

Since the AE-CM builds upon the CM, any implementation also contains an averaging epoch. In case of pre-training, both sub-networks need to be initialized. We favor a straightforward end-to-end training of the DAE (without drop-out or noise) over a few epochs. The clustering module is then initialized using k-means++ on the embedded dataset. Finally, the CM is optimized alone using $\mathcal{L}_{\text{CM}}$ for a few epochs.

5 Experiments

In this section, we evaluate the clustering module and the AE-CM on several data sets covering different types of data. To highlight the generality of our method, we rely only fully connected architecture, ie. we do not use convolution layers even for image data sets. That said, we focus on general purpose baselines. The experiments were conducted on an Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz with 32Gb of RAM supported with a NVIDIA(R) Tesla V100 SXM2 32GB GPU.

5.1 Experimental Setting

5.1.1 Datasets

We leverage eight common benchmark data sets in the deep clustering literature plus one synthetic data set:

- **MNIST** LeCun et al. (1989) contains 70, 000 handwritten images of the digits 0 to 9. The images are grayscale with the digits centered in the 28×28 images. The pixel values are normalized before processing.
- **fMNIST** Xiao et al. (2017) contains 70, 000 images of fashion products organized in 10 classes. The images are grayscale with the product centered in the 28×28 images. The pixel values are normalized before processing.
- **USPS**[1] contains 9298 images of digits 0 to 9. The images are grayscale with size 28×28 pixels. The pixel values are normalized before processing.

[1] https://github.com/XifengGuo/IDEC/files/1613386/usps.zip.

- **CIFAR10** Krizhevsky et al. (2009) contains 60, 000 color images of 10 classes of subjects (dogs, cats, airplanes...). Images are of size 32×32. The pixel values are normalized before processing.
- **Reuters10k**[2], here abbreviated **R10K**, consists of 800, 000 news articles. The dataset is pre-processed as in Guo et al. (2017) to return a subset of 10, 000 random samples embedded into a 2, 000-dimensional space (tf-idf transformation) and distributed over 4 (highly) imbalanced categories.
- **20News**[3] contains 18, 846 messages from newsgroups on 20 topics. Features consists of the tf-idf transformation of the 2000 most frequent words.
- **10 × 73k** Zheng et al. (2017) consists of 73, 233 RNA-transcript belonging to 8 different cell types Jang et al. (2017). The features consists of the log of the gene expression variance of the 720 genes with the largest variance. The dataset is relatively sparse with 40% of entries null.
- **Pendigit** Alimoglu and Alpaydin (1996) consists of 10, 992 sequences of coordinates on a tablet captured as writers write digits, thus 10 classes. The dataset is normalized before processing.
- **5 Gaussians** consists of $N = 2000$ points in $\mathbb{R}^2$ drawn from a mixture of five bi-variate Gaussians ($K = 5$). The dataset is depicted in Fig. 1.

5.1.2 Evaluation metrics

The clustering performance of each model is evaluated using three frequently-used metrics: the Adjusted Rand Index (ARI) (Hubert & Arabie, 1985), the Normalized Mutual Information (NMI) (Estévez et al., 2009), and the clustering accuracy (ACC) (Kuhn, 1955). These metrics range between 0 and 1 where the latter indicates perfect clustering. For legibility, values are always multiplied by 100. For each table, scores not statistically different (t-test $p < 0.05$) from the best score of the column are marked in boldface. The $*$ indicates the model with the best run. A failure ($-$) corresponds to an ARI close to 0.

5.1.3 Baselines

We include two baselines for the CM: k-means (KM), a GMM with full co-variance and an isotropic GMM (iGMM) with uniform mixture weights. The latter differs from the model the clustering module derives but the Dirichlet prior on the responsibilities yields non tractable updates and a Dirichlet prior on the mixture weights harms the performance.

We compare the AE-CM to four baselines reviewed in Sect. 2: DEC (Xie et al., 2016), its extension IDEC (Guo et al., 2017), DCN (Yang et al., 2017) and DKM (Fard et al., 2020). We include as the naive approach (AE+KM) consisting of a trained DAE followed by k-means on the embedding. We also add ClusterGAN (Mukherjee et al., 2019) and VIB-GMM (Uğur et al., 2020) as alternatives based on variational autoencoders (Kingma & Welling, 2013) and generative adversarial networks (Goodfellow et al., 2014), respectively.

Random and pre-trained initialization are indicated with ^{r} and ^{p}, respectively. If omitted, the initialization is random. Every experiment is repeated 20 times.

[2] https://github.com/XifengGuo/IDEC/tree/master/data/reuters.
[3] http://people.csail.mit.edu/jrennie/20Newsgroups.

Table 1 The clustering performance (×100) of different models on the selected datasets

Model	MNIST			fMNIST			USPS			CIFAR10		
	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC
KM^r	37.8	**49.9***	54.5*	36.6	51.6	53.2	52.6	**61.8**	63.0	4.2	8.1	20.8
KM^p	36.9	**49.2**	54.0	35.2	51.0	50.8	50.2	60.8	61.1	4.2	8.1	20.7
GMM^r	23.2	37.8	40.3	34.3	49.3	51.8	35.1	52.5	48.2	**5.0**	9.1	**22.5***
GMM^p	24.8	37.5	42.3	34.3	49.3	52.4	33.0	52.0	45.1	**5.1***	**9.3**	**22.4**
iGMM^r	31.3	42.7	48.5	35.7	50.7	51.4	44.2	55.3	56.2	4.1	7.9	21.1
iGMM^p	31.1	42.7	47.5	35.4	50.8	51.6	44.4	56.1	55.6	4.1	7.8	21.1
CM^r	**39.7***	**50.0**	**56.8**	**42.3**	**54.0**	**62.0**	**54.3**	**62.8**	**67.0***	4.8	8.9	**22.0**
CM^p	**39.1**	**49.5**	**55.9**	**41.4***	**53.4***	**60.7***	**53.4***	**62.8***	63.7	4.9	**9.3***	**22.3**
Model	R10K			20News			10x73k			Pendigit		
	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC
KM^r	**33.8**	38.1	**61.2**	14.8	32.3	31.0	36.5	55.4	55.0	**56.5***	67.8	**71.1**
KM^p	29.5	36.0	58.3	14.8	33.5	32.0	36.7	55.5	55.3	**58.1**	68.9	**72.7**
GMM^r	13.5	18.4	46.8	13.2	34.6	31.6	32.2	50.4	52.4	51.3	68.4*	65.7*
GMM^p	11.8	14.8	47.5	10.7	31.2	27.2	32.0	50.8	51.5	54.3	**69.7**	**71.1**
iGMM^r	**39.8**	**44.2**	**66.9**	**18.4**	**41.8***	**37.4***	34.2	53.7	53.9	**58.1**	**69.0**	**72.7**
iGMM^p	27.3	32.3	60.4	13.4	36.8	31.4	33.1	53.1	52.1	**56.9**	**68.9**	**72.0**
CM^r	**38.5**	**41.0**	**64.9**	9.7	21.2	18.3	**54.8**	**63.8**	**69.5**	**57.3**	67.0	**72.0**
CM^p	32.6*	39.2*	60.2*	16.3*	28.8	30.8	**55.4***	**64.0***	**71.0***	**57.3**	66.9	**72.3**

5.1.4 Implementation

Both CM and AE-CM are implemented using TensorFlow 2.1 (Abadi et al., 2016)[4]. We also re-implemented DEC, IDEC and DCN. All deep models but ClusterGAN and VIB-GMM use the same fully connected autoencoder d-500-500-2000-p-2000-500-500-d and leaky relu activations, where d and p are the input and feature space dimensions, respectively. For ClusterGAN and VIB-GMM, we used the architecture provided in the original code. As well, the DAE reconstruction loss is the mean square error, regardless of the dataset and of the model, except for VIB-GMM on images which requires a cross-entropy loss (it under performs, otherwise). CM and its baselines are trained for up to 150 epochs, deep models for 1000 epochs. The hyper-parameters (batch-size, p, concentration, etc.) are listed in Tables 12 and 13.

5.2 CM: evaluation

Recall that the loss of the clustering module is a lower bound of the objective function of its underlying isotropic GMM which approximates k-means. Moreover, the optimization of

[4] Code available at: https://github.com/Ahcene-B/clustering-Module.

Table 2 Average runtime of each model to cluster MNIST in 150 epochs

Model	KM^r	GMM^r	$iGMM^r$	CM^r
Runtime	1.7s	2m4s	56s	2m44s

Table 3 Clustering performance (ARI) of the CM on the *5 Gaussians* dataset trained with various combination of the terms of its original loss (first line)

E_0	✓		✓	✓	✓	✓	✓	✓	✓
E_1	✓	✓		✓	✓	✓			
E_2	✓	✓	✓		✓		✓		
E_3	✓	✓	✓	✓				✓	
ARI	**83.3**	38.2	45.0	66.8	71.4	50.5	46.2	67.6	59.0

the CM is based on gradient descent instead of EM. We compare these three models as a sanity check, and show that, despite the differences, they report similar clustering performance. We also present an ablation study of the loss and a model selection scheme for the CM. An analysis of the hyper-parameters and of the Dirichlet prior are reported in Appendix 1 and 2, respectively.

5.2.1 CM: clustering performance

In this experiment, we compare clustering performances and initialization schemes. Random and *k*-means++ initializations are indicated with the superscripts ^{r} and ^{p}, respectively. Each experiment is repeated 20 times. We report averages in Table 1. For each dataset, scores not statistically significant different from the highest ($p < .05$) score are marked in boldface. The $*$ indicate the model with the highest best score among its 20 runs. An extended table including standard deviations and best run can be found in Table 11 of Appendix 4. Average runtime for MNIST are reported on Table 2.

As expected, the clustering module performs similarly to iGMM and *k*-means on every dataset with respect to almost every metrics and for any initialization scheme. Often the *k*-means++ initialization does not improve the results. In the case of the clustering module the difference is never significant except on the 20News dataset.

5.2.2 CM: runtimes

We compare here the runtimes of the different method on MNIST. For a fair comparison, we do not use any early-topping criterion and all the methods are run for exactly 150 epochs. We report on Table 2 average over 10 runs.

It appears clearly that EM-based models are much faster. Interestingly GMM is slower than iGMM, although they share the same implementation. This difference is certainly due to the extra computations needed to update the covariance matrices.

5.2.3 CM: ablation study of the loss

The loss function of the CM arises as a whole from the $\mathcal{Q}$ function of the underlying GMM. Nevertheless, for additional insight, we perform here an ablation study of its terms. We train a CM with different combinations of the terms of its original loss (Eq. (7)). To

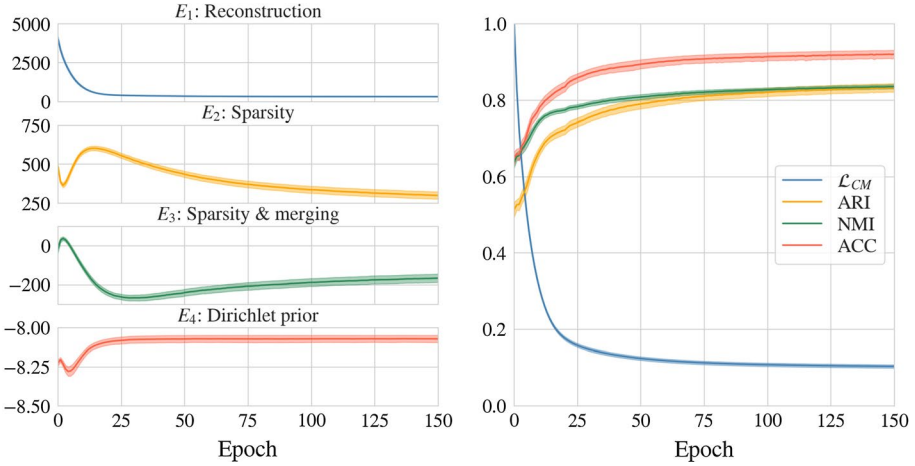

Fig. 3 Evolution of the total loss, each of its term and of the clustering metrics during the training of the CM on the 5 Gaussians dataset

highlight the influence of each term, we focus on the *5 Gaussians* dataset (depicted in Fig. 1). Table 3 reports the clustering performance in terms of ARI.

The gap in terms of ARI between the CM trained with the complete loss (first line) and any other combination of its terms confirms the coherence of the loss. The model trained without the reconstruction term E_0 reports the worse score. The ablation of the single terms indicate that the reconstruction term E_0 is the most important followed by the sparsity term E_1. Although the removal of only E_3 has the least impact, it is the only term that combined with E_0 reports better performance than a loss function made of solely the reconstruction term E_0.

Figure 3 illustrates the behavior of each term of the loss of the CM ($\mathcal{L}_{CM}$) during the training. Given the variations of each curves, it seems that during the first 25 epochs the optimization focuses on minimizing the reconstruction term even if it implies an increase of the other ones. However, the complete loss curve does not flatten out until E_3 reaches its minimum. From there, a second phase begins where the total loss and the clustering metrics slowly grow in opposite directions. Interestingly, as the metrics increase and the clustering improves, E_3 also increases, which is contrary to the expected behavior. On the other hand, E_2 which is of the same magnitude as E_3 and is also influenced by the sparsity of the assignments, continuously decreases without reaching a minimum. Overall, these curves suggest that both E_2 and E_3 could be used to either stop early the training or selecting the best run.

5.2.4 CM: model selection

In an unsupervised scenario, true labels are not available. It is thus necessary to have an internal measure to avoid selecting a sub-optimal solution.

There are two natural choices: select either the clustering associated with the lowest loss or the less ambiguous clustering. In the first case, the sparsity of the clusters responsibilities $\boldsymbol{\gamma}_i$ might be eclipsed by other aspects optimized by the $\mathcal{L}_{\text{CM}}$, such as the reconstruction term. On the other hand, by selecting only given the sparsity, we may end up always

Table 4 Adjusted Rand index of the run with lowest $\mathcal{L}_{\text{sp}}$, the average run and the best run

Criterion	MNIST			fMNIST			USPS			CIFAR10		
	$\mathcal{L}_{\text{sp}}$	Avg.	Best	$\mathcal{L}_{\text{sp}}$	Avg.	Best	$\mathcal{L}_{\text{sp}}$	Avg.	Best	$\mathcal{L}_{\text{sp}}$	Avg.	Best
CM^r	**42.7**	39.7	43.5	**44.2**	42.3	44.6	**54.8**	54.3	58.6	**4.8**	4.79	5.07
CM^p	**43.3**	39.1	43.5	**44.0**	41.4	44.7	**55.8**	53.4	59.2	**4.94**	4.94	5.22
Criterion	R10K			20News			10×73k			Pendigit		
	$\mathcal{L}_{\text{sp}}$	Avg.	Best	$\mathcal{L}_{\text{sp}}$	Avg.	Best	$\mathcal{L}_{\text{sp}}$	Avg.	Best	$\mathcal{L}_{\text{sp}}$	Avg.	Best
CM^r	**56.0**	38.5	56.0	**9.92**	9.74	10.9	54.3	54.8	57.4	57.0	57.3	60.5
CM^p	**62.9**	32.6	62.9	13.1	16.3	22.0	54.9	55.4	62.4	56.4	57.3	60.1

Score larger than the average are marked in boldface

choosing the most degenerate clustering. Leaning on the analysis of Fig. 3, we propose to use $\mathcal{L}_{\text{sp}} = E_2 + E_3$ which sums both terms of the loss governing the sparsity of the $\boldsymbol{\gamma}_i$s but also involves the norm of the $\boldsymbol{\mu}_k$s.

In Table 4, we report the ARI of the runs with the lowest $\mathcal{L}_{\text{sp}}$ for each dataset. For comparison, we also report the average and the largest ARI. Scores selected according to $\mathcal{L}_{\text{sp}}$ that were higher than the average are marked in boldface.

A model selection based on $\mathcal{L}_{\text{sp}}$ finds the best runs only for R10K. Nevertheless, it selects runs with ARI greater than the average in more than half of the cases. In the other cases, the difference to the average score remains below 1 point of ARI expect for CM^p on 20News. The average absolute difference with the best score is 1.60 and 3.10 for CM^r and CM^p, respectively. Without 20News, on which CM^p performs the worst, that average difference drops to 2.25 for CM^p. These are satisfying results that substantiates our heuristic that $\mathcal{L}_{\text{sp}} = E_2 + E_3$ can be used as an internal metric for the CM.

5.3 AE-CM: evaluation

In this section, we compare the clustering performance of our novel deep clustering model AE-CM against a set of baselines. We study the robustness of the model with respect to the number of clusters and a model selection scheme. We also evaluate the quality of the embeddings through the k-means clusterings thereof. Finally, we review the generative capabilities of our model.

5.3.1 AE-CM: clustering performance

We compare now clustering performances and initialization schemes for representative deep clustering models. We reports average ARI, NMI and ACC over 20 runs in Table 5. An extended table including standard deviations and best run can be found in Table 15 of Appendix 3.

In their original papers, the DEC, IDEC and DCN are pre-trained (^{p}). We report here slightly lower scores that we ascribe to our implementation and slightly different architectures. Nonetheless, the take-home message here is the consistency of their poor results when randomly initialized. This reflects an inability to produce cluster from

Table 5 The clustering scores (×100) of representative deep clustering models on the selected datasets

Model	MNIST			fMNIST			USPS			CIFAR10		
	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC
AE+KM	65.6	71.5	78.6	39.0	55.6	53.0	57.1	64.6	67.5	3.2	6.5	18.9
DCN^r	10.1	25.6	25.4	17.0	33.5	29.0	17.9	36.5	37.9	3.2	5.9	18.0
DCN^p	75.6	**82.5**	83.1	38.6	57.1	53.1	63.9	73.1	72.5	0.1	0.6	10.7
DEC^r	11.1	19.0	28.9	22.9	38.1	39.2	36.3	46.9	46.8	3.1	5.7	18.6
DEC^p	73.8	79.0	83.1	41.9	58.6	54.8	**70.0**	**78.1**	**76.3**	3.1	5.6	18.2
IDEC^r	27.5	39.0	42.5	35.2	50.8	48.1	41.8	53.2	54.0	2.2	3.6	14.0
IDEC^p	74.9	80.1	83.4	42.8	59.8	55.4	**70.0**	**78.0**	**76.1**	4.2	7.4	20.2
DKM^r	72.5	77.3	81.2	41.8	56.4	54.6	58.3	67.0	68.6	**5.8***	**9.9***	21.3
DKM^p	74.0	78.3	82.7	36.2	52.0	47.0	60.4	71.8	68.9	**5.9**	**10.0**	19.7
ClusterGAN^r	63.6	71.8	76.8	**46.5**	**60.7**	**59.0**	57.4	67.9	70.0	3.2	7.6	20.4
VIB-GMM^r	73.3*	78.3*	81.5*	43.7*	58.4*	**59.3***	59.9	67.7	68.4	**6.0**	**10.1**	**24.0***
AE-CM^r	**77.9**	80.9	**86.1**	43.7	55.6	**59.2**	55.1	63.4	65.8	4.1	7.5	20.4
AE-CM^p	**79.4**	**82.4**	**86.5**	43.1	56.3	**58.5**	**69.7***	76.7*	**76.8***	4.1	7.6	20.2
Model	R10K			20News			10x73k			Pendigit		
	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC
AE+KM	61.0*	56.8	74.5*	11.3	27.4	24.8	54.3	72.5	64.4	55.2	68.2	70.2
DCN^r	18.0	19.3	49.5	0.0	0.2	5.6	5.3	17.2	23.9	0.1	0.8	10.8
DCN^p	**65.4**	**61.1**	**76.6**	11.7	33.5	25.3	9.6	13.8	25.2	56.8	72.0	70.8
DEC^r	12.2	13.2	43.8	3.2	7.8	10.0	31.4	43.5	46.3	36.8	52.3	49.3
DEC^p	56.8	56.0	72.8	5.5	11.3	11.8	53.5	67.1	62.1	59.6	72.8	72.2
IDEC^r	8.6	9.5	44.1	0.0	0.1	5.5	33.7	46.5	44.4	43.3	61.2	53.9
IDEC^p	59.7	56.3	73.9	5.9	12.6	12.0	60.1	75.9	66.5	57.9	71.6	71.0
DKM^r	51.3	49.5	72.3	4.7	14.1	10.9	65.5	71.3	77.0	52.4	65.6	66.9
DKM^p	57.7	55.5	**76.5**	20.9	39.2	34.3	38.1	55.4	51.6	15.4	27.4	25.1
ClusterGAN^r	33.7	35.5	61.4	18.6	34.1	34.1	39.5	52.1	55.5	**62.9**	74.2	**75.6**
VIB-GMM^r	27.8	28.7	56.6	0.0	0.0	0.0	51.5	60.7	60.0	**64.5**	**75.7**	**74.2**
AE-CM^r	42.9	45.6	67.7	**31.5***	**45.3***	**43.3***	73.1	79.0	80.4	**64.6***	**75.0***	**75.7***
AE-CM^p	64.1	60.0*	**76.3**	16.8	29.0	32.5	**82.3***	**83.7***	**89.4***	**60.1**	**70.4**	**73.0**

scratch, regardless of implementation. Note that, in that case, even k-means outperforms all of them on all the datasets (see Table 1). On the other hand, DKM does return competitive score for both initialization scheme. Such results yield the question of how much clustering that is actually performed by DEC, IDEC and DCN and how much that is due to the pre-training phase.

In practice, DKM has proven sensitive to the choice of its λ hyper-parameter and to the duration of the optimization. For example, we could not find a value able to cluster Pendigit. We conjecture that expanding the clustering term of DKM's loss, as we did between Eqs. (1) and (3), would improve the robustness of the model.

On six of the datasets, at least one of the variants of AE-CM reports the highest average or highest best run. Especially, AE-CM^r produces competitive clusterings on all datasets except CIFAR10 despite its random initialization. This setback is expected as

Table 6 Average runtime of each model to cluster MNIST in 150 epochs

Model	DEC^r	IDEC^r	DCN^r	DKM^r	ClusterGAN^r	VIB-GMM^r	AE-CM^r
Runtime	25m13s	27m51s	39m18s	38m29s	3h17m30s	29m26	19m28

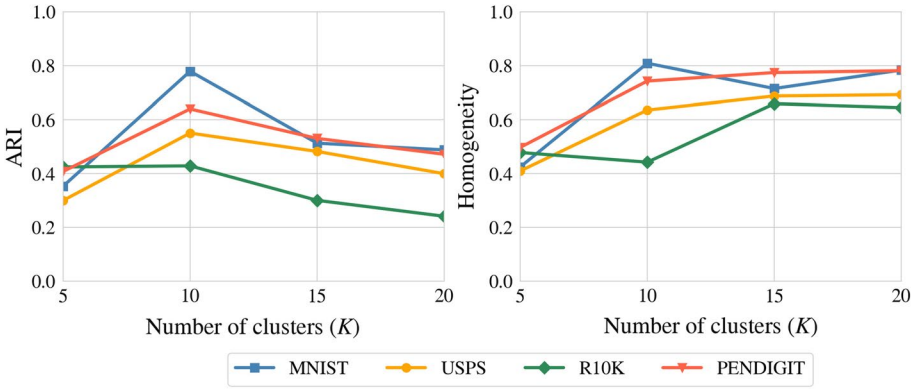

Fig. 4 Adjusted Rand index and homogeneity score versus number of clusters. The correct number of clusters being $K = 10$

clustering models are known to fail to cluster color images from the raw pixels (Jiang et al., 2016; Hu et al., 2017).

Also our AE-CM with random initialization always surpasses AE+KM except on R10K and CIFAR10, and even outperforms all the competitors by at least 20 ARI points on 20News. On the down side, AE-CM^r is associated with large standard deviations which implies a less predictable optimization (see Table 15 of Appendix 3). Therefore, we investigate an internal measure to select the best run.

5.3.2 AE-CM: runtimes

We compare here the runtimes of the different method on MNIST. For a fair comparison, all the methods use the same batch size of 256 instances. We report on Table 6 average over 10 runs. We do not used any early-stopping criterion.

Note that our implementation of DEC, IDEC and DCN are based on that of AE-CM. Hence these are the most comparable. The advantage goes to the model joint optimization, AE-CM, which is more that 5 min faster. ClusterGAN is the slowest method. It also has the most complex architecture.

5.3.3 AE-CM: robustness to the number of clusters

In the previous experiments, we provided the true number of clusters to all algorithms for all datasets. In this experiment, we investigate the behavior of the AE-CM when it is set with a different number of clusters on four datasets: MNIST, USPS, R10K and Pendigit. Figure 4 shows the evolution of the ARI (left) and the homogeneity (Rosenberg &

Table 7 Adjusted Rand index of the run with lowest $\mathcal{L}_{sp}$, the average run and the best run

Criterion	MNIST			fMNIST			USPS			CIFAR10		
	$\mathcal{L}_{sp}$	Avg.	Best	$\mathcal{L}_{sp}$	Avg.	Best	$\mathcal{L}_{sp}$	Avg.	Best	$\mathcal{L}_{sp}$	Avg.	Best
AE-CM^r	**88.6**	77.9	88.6	**47.8**	43.7	48.9	**55.4**	55.1	60.6	**5.29**	4.15	5.29
AE-CM^p	**80.3**	79.4	80.3	37.3	43.1	48.4	61.0	69.7	80.3	2.97	4.13	5.56
Criterion	R10K			20News			10x73k			Pendigit		
	$\mathcal{L}_{sp}$	Avg.	Best	$\mathcal{L}_{sp}$	Avg.	Best	$\mathcal{L}_{sp}$	Avg.	Best	$\mathcal{L}_{sp}$	Avg.	Best
AE-CM^r	36.7	42.9	62.7	**36.7**	31.5	38.7	72.8	73.1	85.6	**64.6**	64.0	69.5
AE-CM^p	**64.8**	64.1	66.7	14.6	16.8	21.2	79.9	82.3	86.9	**66.3**	65.5	70.5

Scores larger than the average are marked in boldface

Table 8 Average clustering performance of k-means on different embeddings.

	KM	AE+KM	DCN+KM	DEC+KM	IDEC+KM	DKM+KM	AE-CM+KM
ARI	37.8	65.6	64.5	11.1	27.5	**74.5**	**75.2**
NMI	49.9	71.5	71.1	19.0	38.9	**78.8**	**78.5**
ACC	54.5	78.6	76.2	28.9	42.5	**83.0**	**84.8**

Hirschberg, 2007) score (right). The latter measures the purity in terms of true labels of each cluster. The number of cluster varies from 5 to 20. The correct value for all datasets is 10.

The ARI curves (left plot) reach their maximum at 10 and then decrease. This behavior is expected since this metric (as well as NMI and ACC) penalizes the number of clusters. On the other hand, the homogeneity curves (right plot) increase with K and stabilize for K larger than 10. The convergence of these curves indicates that the clustering performance of the AE-CM do not degrade if K is set larger than the ground truth. Such a results suggests that, when K is larger than the ground truth, the AE-CM finds solutions that are partitions of those found with smaller K. Such a phenomenon is illustrated in Appendix 3.

5.3.4 AE-CM: model selection

Similarly to the CM, we discuss here a model selection heuristic for the AE-CM. The rationale behind the use of a DAE is to have an encoding facilitating the objective of the clustering module. Hence, we propose to use the heuristic of the CM (Sect. 5.2.4). In Table 7, we report the ARI of the runs with the lowest $\mathcal{L}_{sp}$ for each dataset. For comparison, we also report the average and best ARI. Selected scores greater than the average are marked in boldface.

Again, scores associated to the lowest $\mathcal{L}_{sp}$ are better than the average more than half of the time. The criterion detects the best runs of AE-CM^r on MNIST and CIFAR10

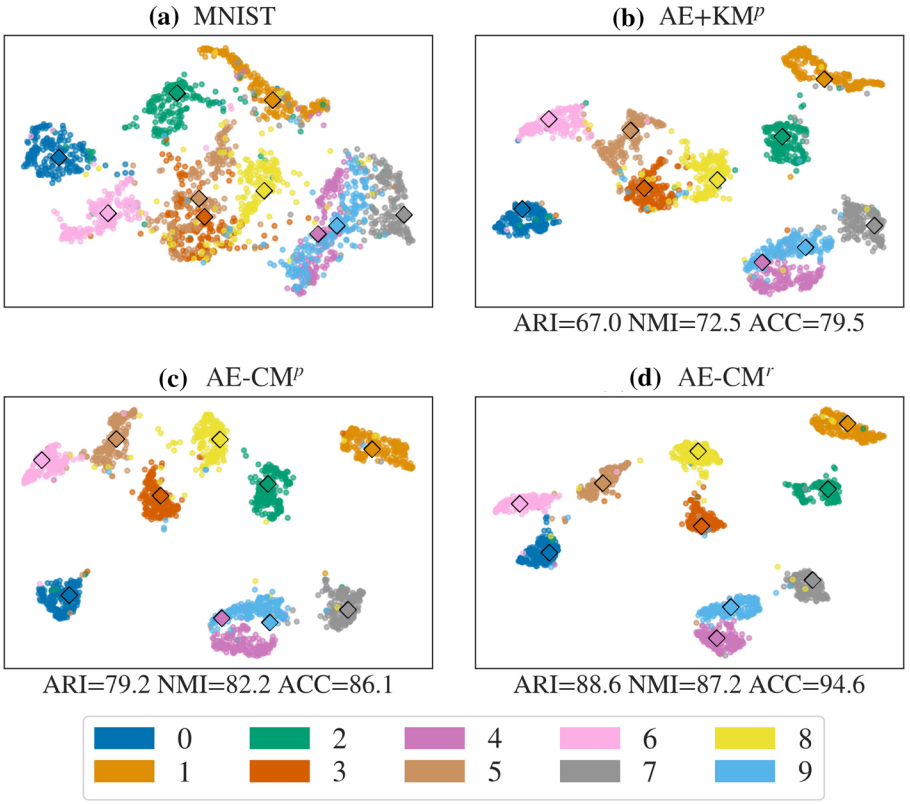

Fig. 5 UMAP representation of a subset of MNIST and embeddings thereof learned by AE, IDEC and AE-CM. The squares indicate the centroids

and of AE-CM^p on MNIST. The average absolute difference to the highest score is 6.5 and 6.6 ARI points for AE-CM^r and AE-CM^p, respectively. In summary, although $\mathcal{L}_{\text{sp}}$ as a criterion does not necessarily select the best run, it filters out the worst runs.

5.3.5 AE-CM: embeddings for *k*-means

All baselines, including our model, are non-linear extensions of *k*-means and aim to improve the AE+KM. We audit the methods by running *k*-means on the embeddings produced by the 20 runs with random initialization on the MNIST dataset computed for Table 5 and report the average ARI, NMI and ACC in Table 8.

First, *KM* reports the worse results. This means that applying *k*-means on a feature space learned by an autoencoder does improve the quality of the clustering. Next, the results clearly show the superiority of methods utilizing a joint optimization, i.e., DKM^r +KM and our AE-CM^r+KM. Interestingly, the scores of DCN^r+KM are here better than those of DCN^r. This discrepancy is certainly due the moving average used to update the centroids.

We continue the analysis of the embeddings with UMAP McInnes et al. (1802) projections. Figure 5 depicts the projections of different embeddings of the same 2000

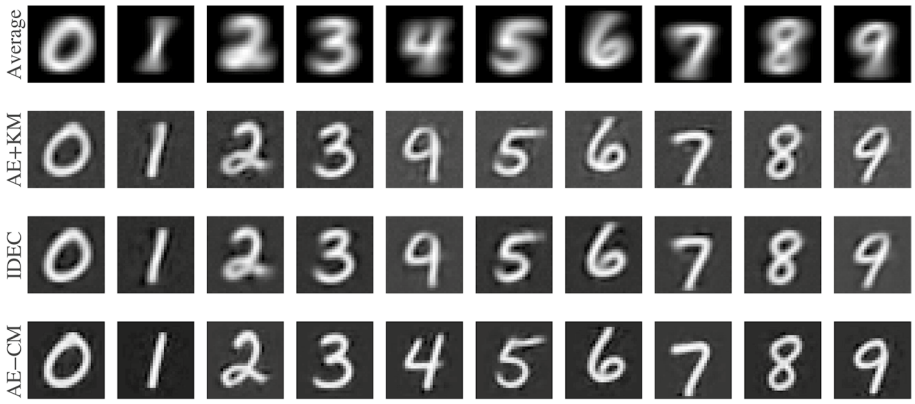

Fig. 6 Centroids mapped back to image space for AE+KM, IDEC^p, DKM^r, and AE-CM^r. The first row displays the average image of each class

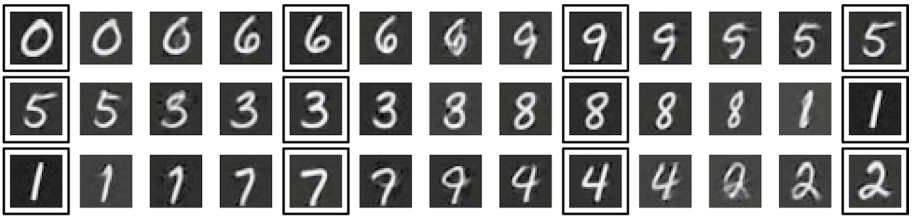

Fig. 7 Linear interpolations between different centroids (plots with border) produced with the AE-CM

data-points. Figure 5a represents the UMAP projection from the input space. For (5b), we used the best run of the AE+KM. For consistency, we used that embedding for the AE-CM^p. Hence, Fig. 5c does not show the best run the AE-CM^p. Finally (5d) is based on the best run of the AE-CM^r.

The projection of MNIST from the input space (5a) has two pairs of classes entangled: (3, 5) and (4, 9). The end-to-end training of the DAE (5b) successfully isolates each class except for cluster 4 (dark pink) and cluster 9 (light blue) which stay grouped together, although separable. The AE-CM^p (5c) further contracts the cluster around the centroids found by the AE+KM, but fails to separate 4 and 9. Remark that even the best run of the AE-CM^p does not to correctly split the data points. The centroids for 4 and 9 in Fig. 5b and c are in comparable positions: they align along the gap separating the true clusters. This suggests that the optimization of the AE-CM^p did not move them much. This remark applies to the pre-trained baselines, as well. Lastly, the AE-CM^r successfully produces homogeneous groups (5f). Remark that the original entanglements of the pairs (3, 5) and (4, 9) are suggested by the trails between the corresponding clusters.

The previous observations summarize into two insights on the behavior of the AE-CM. If the AE-CM starts with an embedding associated to a low reconstruction loss for the DAE, the optimization contracts the clusters which yields higher ARI

scores. However, it is unable to move the centroids to reach another local optimum. Although the AE-CM^r separates (4, 9), it also produces clusters more spread than those of the AE-CM^p. The improved performances of the latter over AE+KM indicates that the AE-CM^r would benefit from tighter groups.

5.3.6 AE-CM: sample generation and interpolation

Thanks to the reversible feature maps obtained by the DAE, both the AE-CM and its baselines (except DEC) are generative models. Figure 6 shows the decoding of the centroids of the best run of AE+KM (ARI=67.7), IDEC^p (ARI=77.2), DKM^r (ARI=83.6) and AE-CM^r (ARI=88.6). AE+KM's and IDEC^p's centroids for the 4 and 9 both look like 9's. With an ARI and an ACC larger than 80 and 90, respectively, DKM^r and AE-CM^r both clustered the data correctly and found correct centroids for each class. Both models produce clear images for each class, which align reasonably well with the washed-out average image of the respective classes (first row).

Being a generative model, the AE-CM can also be used to interpolate between classes. Figure 7 shows a path made of nine interpolations between the ten centroids of the AE-CM^r. We observe smooth transitions between all the pairs, which indicate that the model learned a smooth manifold from noise (random initialization).

6 Conclusion

We presented a novel clustering algorithm that is jointly optimized with the embedding of an autoencoder to allow for nonlinear and interpretable clusterings. We first as a key result showed that the objective function of an isotropic GMM can be turned into a loss function for autoencoders. The clustering module (CM), defined as the smallest resulting network, was shown to perform similarly to its underlying GMM in extensive empirical evaluations.

Importantly, we showed how the clustering module can be straightforwardly incorporated into deep autoencoders to allow for nonlinear clusterings. The resulting clustering network, the AE-CM, empirically outperformed existing centroid-based deep clustering architectures and performed on par with representative contemporary state-of-the-art deep clustering strategies. Nevertheless, the AE-CM, and to a lesser extent of the clustering module itself, presented a greater volatility when trained from a randmly initialized network. We expect that we could improve on that point by involving an annealing strategy on the parameter, similarly to what is done in DKM and VIB-GMM.

A future line of work consist of extending the panel of deep architectures into which the clustering module can be nested. In order to improve performance on image data sets, especially, it is necessary to involve convolution. However, standard image-specific architectures are not structured as autoencoder. This raises the question of the robustness of our model with respect to the symmetry of the DAE, especially for applications where the computation of class representative is not a must.

From a theoretical point of view, we believe that the derivations that led to the neural interpretation of Gaussian mixture models could benefit other mixture models such as the von Mises-Fisher mixture models (Hasnat et al., 2017) or hidden Markov models (HMM). The case of Gaussian-HMM seems especially promising as it allows to bridge with Recurrent networks (Salaün et al., 2019).

Table 9 The clustering performance (×100) of different models on the selected datasets

Model	Breast			Ecoli			Glass		
	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC
iGMM^r	40.0	51.4	48.3	50.6	64.2	64	21.8	41.7	44.9
DEC^r	77.8	66.7	86.7	48.8	49.5	63.6	22.8	31.9	47.5
IDEC^r	50.7	44.6	68.6	35.2	41.4	55.2	17.9	30.4	42.3
DCN^r	72.7	57.2	79.7	45.7	49.9	63.4	20.2	32.9	43.1
DKM^r	83.1	73.2	95.6	51.2	61.5	64.3	23.2	37.9	49.3
CM^r	35.9	48.4	47.5	66.7	65.6	76.0	27.1	43.2	49.1
AE-CM^r	72.9	58.0	72.8	72.4	68.1	76.7	37.8	47.3	59.8
Model	Iris			Wine			Yeast		
	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC
iGMM^r	62.0	65.9	83.3	76.9	81.3	83.7	15.7	28.8	38.8
DEC^r	37.5	44.3	61.6	23.8	28.7	55.9	9.1	16.9	31.6
IDEC^r	26.6	33.4	60.0	24.8	28.3	57.8	7.6	15.9	31.1
DCN^r	45.3	53.2	67.7	41.3	46.7	65.7	6.8	15.4	37.4
DKM^r	58.4	65.9	77.2	72.4	70.6	90.2	17.5	29.6	42.4
CM^r	59.1	63.4	81.5	82.0	82.2	89.6	15.5	28.5	38.0
AE-CM^r	78.9	85.0	94.4	86.0	85.4	93.0	17.7	27.1	46.6

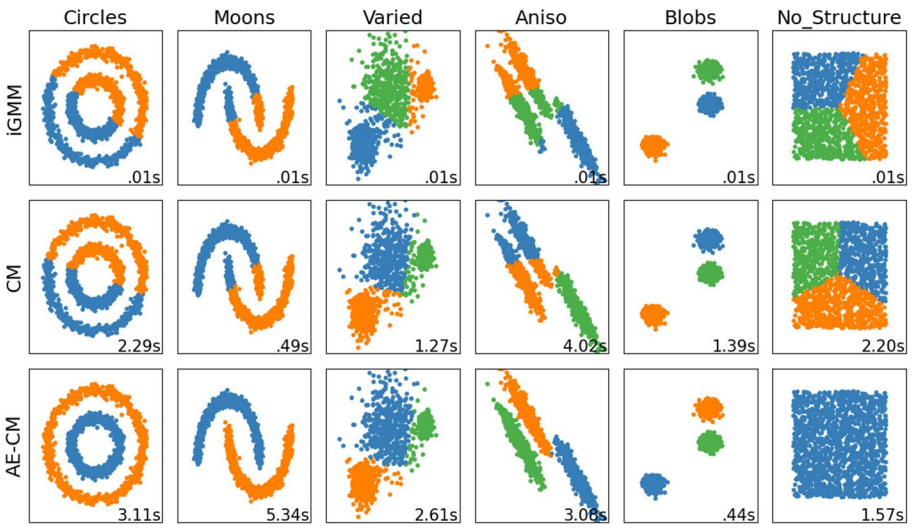

Fig. 8 Clustering of six toy datasets by iGMM, CM and AE-CM

Table 10 Hyper-parameters used for clustering the toy datasets

Dataset	α	β	λ	Encoder Archi.
Moons	11	100	0.001	2-20-20-20-1
Circles	11	0.001	0.001	2-quad-3
Varied	11	100	0.001	2-3
Aniso.	11	1	0.001	2-3
Blobs	11	1	0.001	2-3
No structure	0.1	1	0.001	2-3

Appendix 1: additional experiments

UCI datasets

In this section, we report clustering performance of iGMM, DEC, IDEC, DCN and DKM as well as our models CM and AE-CM on a selection of UCI datasets. We consider here only random initialization. Deep clustering models share the same architecture: $d - 2 \times k - d$, where d is the dimension of the input and k the number of clusters. The k-means clusterings of DEC and IDEC are updated every 5 epochs and $\gamma = 0.1$. The λ hyperparameter of DCN and DKM are set to 1 and 0.001, respectively. For CM and AE-CM the three hyperparameters (α, β, λ) were leaned using Bayesian optimization. We report average ARI, NMI and ACC over 10 runs (Table 9).

Scikit-learn benchmark

Figure 8 reports clusterings of the scikit-learn toy datasets[5]. We compare here, iGMM, CM and AE-CM. The CM is optimized using SGD while the AE-CM relies on Adam. Training are stopped if the difference between the clusterings of two successive iterations are difference by less that 0.1%. The models are run 10 times for up to 100 epochs with a fixed batch size of 20 instances. The average run time is shown in the lower right corner of each plot (Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz). The parameters for each dataset are given in Table 10.

The standard way to split the two circles and the two moons datasets is to use a polynomial kernel of degree 2 and at least 3 respectively.

To highlight the relationship between embedding and feature maps, we choose for these datasets specific architectures of the deep autoencoder of AE-CM. For the two moons dataset, we want the AE-CM to learn an embedding function approximating a polynomial of degree at least. Therefore, we use 3 layers with 20 units followed by a layers with a single unit. The decoder is the mirror of the encoder. For the two circles dataset, the encoder consists of a quadratic layer, i.e.

$$(x_1, x_2) \mapsto (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2, x_2 x_1),$$

[5] https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html.

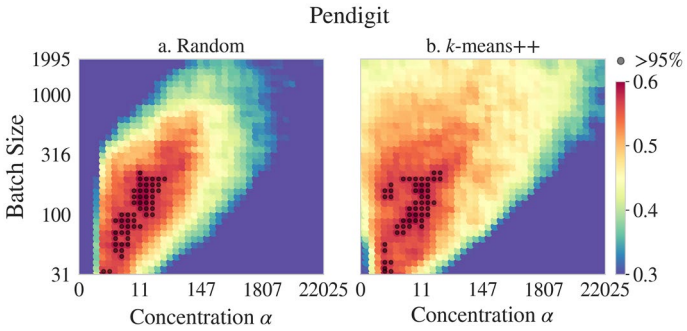

Fig. 9 Clustering performance (ARI) for different combinations of batch-size, concentration and initialization scheme. Black dots indicate average ARI greater than the ones reported in Table 1

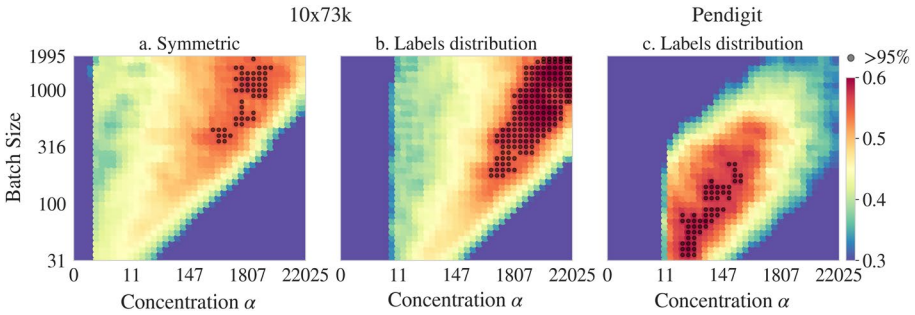

Fig. 10 Clustering performance (ARI) for different combinations of batch-size, concentration and prior distribution. Black dots indicate average ARI greater than the ones reported in Table 1

followed by a dense layer with a dimension 3 output. That way, the function associated to the encoder is a feature for a polynomial kernel of degree 2. The training just has to find the proper weights of the quadratic function. As for the decoder, it consists of a single layer. The AE-CM successfully clustered the two circles and two moons dataset, suggesting that it indeed learned embedding functions associated to polynomial kernels of degree 2 and at least 3, respectively.

Finally, the last dataset consisting of a square filled with a single class, we chose an α lower than 1 for both CM and AE-CM. Such a setting, informs the model that the three classes will be very imbalanced. both models reacted differently. Indeed, only AE-CM was able to perfectly assign all the points to a single cluster.

Appendix 2: supplementary materials related to the clustering module

CM: hyper-parameters

The clustering module depends on two hyper-parameters: the size of the mini-batches and the concentration of the Dirichlet prior. To visualize their influence on the clustering performance, we trained a CM on Pendigit with various sizes of batch and concentrations.

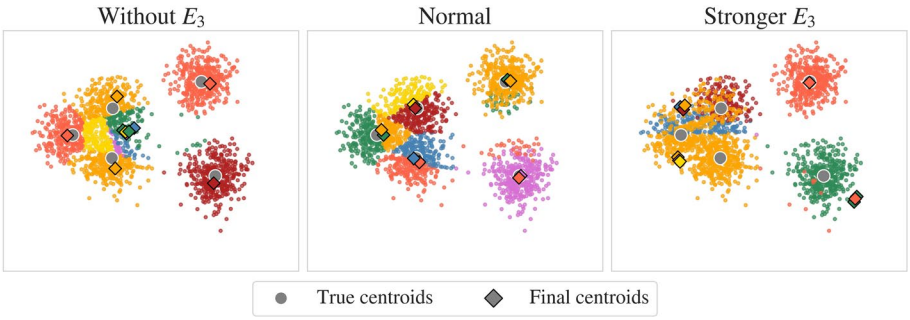

Fig. 11 Final positions of the centroids depending on the importance of E_3 in the loss of the clustering module

Figure 9 shows the variation of the final ARI score for both a random (left) and a k-means++ initialization (right). Both axes' scales are logarithmic: exponential base for the x-axis and base 10 for the y-axis. Each combination is run once. The ARI of each dot is the average of the nine neighboring combinations. Black dots indicate an average ARI greater than the ones reported in Table 1.

There is a lower bound on α under which the optimization of a randomly initialized model underperforms or fails. The k-means++ initialization removes this border and spreads out the well-performing area. The distribution in both settings means that the hyper-parameters can be tuned by fitting a bi-variate Gaussian distribution.

CM: asymmetric prior

So far we considered only symmetric Dirichlet priors ($\boldsymbol{\alpha} = \alpha \mathbf{1}_K$, $\alpha \in \mathbb{R}^+$) regardless of the imbalance between the labels. Here, we repeat the previous experiment using the true labels distribution as the prior, i.e. $\boldsymbol{\alpha} = \alpha \boldsymbol{f}$ where $\boldsymbol{f} \langle f_k \rangle_K \in \mathbb{S}^K$ is the frequency of each label. In terms of implementation, E_4 is computed by sorting both $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}_i$. We evaluate results on the 10×73k and Pendigit datasets, which have unbalance and balanced classes, respectively. We consider here only random initializations. Again, black dots indicate an average ARI greater than the ones reported in Table 1.

Figure 10b contains more black dots and a larger red area compared to 10a. The changes are greater than between Figs. 10c and 9b. This discrepancy between the datasets illustrates that unbalanced ones benefit more from a custom prior. However, a higher concentration is needed to enforce the distribution: the lower bound on α is higher in Fig. 10b and c than in 10a and 9a, respectively. Using the true class distribution, especially if the data is unbalanced, does ease the hyper-parameter selection. Nevertheless, such an information is not always known a priori.

CM: merging clusters with E_3

We claimed in Sect. 3.2 that E_3 favors the merging of clusters. To illustrate this phenomenon, we train CM^r on the 5 Gaussians dataset with twice the number of true clusters (i.e., $K = 10$). We compare three variants of CM's loss function: without E_3, with E_3 and with E_3 multiplied by 1.5. The final centroids and clustering are depicted in Fig. 11. For legibility, overlapping centroids are slightly shifted using a Gaussian noise.

Table 11 The clustering results of the methods on the experimental datasets.

Model		MNIST			fMNIST			USPS			CIFAR10		
		ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC
KM^r	avg std	37.8 ±2.7	**49.9** ±1.9	54.5 ±4.3	36.6 ±1.7	51.6 ±1.2	53.2 ±3.4	52.6 ±2.3	**61.8** ±1.4	63.0 ±3.2	4.2 ±0.1	8.1 ±0.2	20.8 ±0.4
	max	43.4	54.1	62.4	38.5	53.1	59.4	56.2	63.9	68.3	4.3	8.3	21.7
KM^p	avg std	36.9 ±1.7	**49.2** ±1.3	54.0 ±3.0	35.2 ±1.6	51.0 ±1.0	50.8 ±3.4	50.2 ±4.9	60.8 ±2.5	61.1 ±5.2	4.2 ±0.1	8.1 ±0.2	20.7 ±0.3
	max	40.8	52.1	58.8	38.5	53.3	57.9	54.5	63.7	67.2	4.4	8.5	21.5
GMM^r	avg std	12.0 ±1.6	20.9 ±2.1	29.9 ±1.7	26.8 ±2.9	44.2 ±2.3	39.5 ±2.9	9.5 ±4.0	17.6 ±3.9	27.8 ±3.1	0.7 ±0.0	0.9 ±0.0	12.1 ±0.1
	max	15.8	25.4	34.2	31.2	48.4	45.1	19.0	26.7	34.5	0.8	1.0	12.2
GMM^p	avg std	22.7 ±1.7	35.6 ±1.2	42.6 ±1.9	34.6 ±2.2	52.6 ±1.5	47.2 ±2.5	35.1 ±4.4	50.9 ±2.7	52.0 ±3.7	3.4 ±1.0	6.9 ±1.0	20.2 ±0.8
	max	25.2	37.7	45.3	38.3	54.9	49.8	39.8	53.7	56.6	4.5	8.1	21.1
$iGMM^r$	avg std	31.3 ±1.2	42.7 ±0.9	48.5 ±1.6	35.7 ±1.5	50.7 ±0.8	51.4 ±2.7	44.2 ±2.6	55.3 ±2.2	56.2 ±2.8	4.1 ±0.1	7.9 ±0.2	21.1 ±0.3
	max	32.4	43.5	50.3	37.6	51.9	55.0	47.5	58.0	59.7	4.2	8.1	21.6
$iGMM^p$	avg std	31.1 ±1.3	42.7 ±0.7	47.5 ±2.6	35.4 ±1.8	50.8 ±1.1	51.6 ±2.7	44.4 ±1.8	56.1 ±1.3	55.6 ±2.5	4.1 ±0.1	7.8 ±0.2	21.1 ±0.3
	max	32.4	43.5	50.3	37.7	52.1	54.5	47.6	58.1	59.7	4.2	8.1	21.5
CM^r	avg std	**39.7** ±1.9	**50.0** ±1.5	**56.8** ±2.1	**42.3** ±3.4	**54.0** ±2.3	**62.0** ±4.1	**54.3** ±2.1	**62.8** ±1.0	**67.0** ±2.5	4.8 ±0.2	8.9 ±0.3	**22.0** ±0.5
	max	43.5	53.0	60.8	44.6	55.7	64.4	58.6	65.1	75.0	5.1	9.4	23.2
CM^p	avg std	**39.1** ±1.4	**49.5** ±1.1	**55.9** ±1.3	**41.4** ±4.1	**53.4** ±2.8	**60.7** ±5.1	**53.4** ±4.1	**62.8** ±2.3	63.7 ±4.3	**4.9** ±0.1	**9.3** ±0.2	**22.3** ±0.5
	max	43.5	52.9	59.8	44.7	55.8	64.5	59.2	66.9	69.6	5.2	9.7	23.2
Model		R10K			20News			10x73k			Pendigit		
		ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC
KM^r	avg std	**33.8** ±14.3	38.1 ±10.2	**61.2** ±11.9	14.8 ±1.3	32.3 ±1.9	31.0 ±1.9	36.5 ±0.8	55.4 ±0.7	55.0 ±1.8	**56.5** ±3.7	67.8 ±1.9	**71.1** ±4.8
	max	61.5	56.4	80.3	17.4	36.1	34.5	37.3	56.1	56.3	62.3	70.6	76.8
KM^p	avg std	29.5 ±15.1	36.0 ±9.2	58.3 ±10.3	14.8 ±1.5	33.5 ±2.4	32.0 ±2.5	36.7 ±0.3	55.5 ±0.4	55.3 ±1.2	**58.1** ±3.1	68.9 ±1.2	**72.7** ±3.5
	max	61.5	56.5	80.3	17.6	37.9	36.4	37.3	56.0	56.3	62.2	70.6	76.7
GMM^r	avg std	0.1 ±0.1	0.1 ±0.1	26.4 ±0.5	0.1 ±0.0	0.7 ±0.1	7.0 ±0.1	26.3 ±6.7	49.0 ±8.4	42.5 ±6.3	54.4 ±4.5	68.1 ±3.4	66.6 ±6.6
	max	0.3	0.3	27.1	0.2	0.8	7.3	34.2	60.5	51.4	63.2	74.9	79.0

Table 11 (continued)

Model		R10K			20News			10x73k			Pendigit		
		ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC
GMM^p	avg std	29.7 ±15.0	36.2 ±9.2	58.4 ±10.2	14.8 ±1.5	33.5 ±2.4	32.0 ±2.5	38.4 ±2.3	61.7 ±1.5	55.6 ±3.3	55.4 ±3.1	**70.1** ±1.7	**71.5** ±3.0
	max	61.5	56.5	80.3	17.6	37.9	36.4	40.3	62.7	58.1	60.4	73.8	78.6
$iGMM^r$	avg std	**39.8** ±5.2	**44.2** ±3.2	**66.9** ±3.8	**18.4** ±1.1	**41.8** ±1.8	**37.4** ±1.7	34.2 ±0.5	53.7 ±0.9	53.9 ±1.3	**58.1** ±3.8	69.0 ±1.6	**72.7** ±4.5
	max	52.4	51.0	73.8	20.5	45.8	41.2	34.8	55.5	54.8	62.0	70.9	77.3
$iGMM^p$	avg std	27.3 ±17.6	32.3 ±17.5	60.4 ±13.6	13.4 ±1.9	36.8 ±2.7	31.4 ±2.7	33.1 ±4.0	53.1 ±4.2	52.1 ±3.8	**56.9** ±4.3	**68.9** ±2.1	**72.0** ±4.8
	max	45.5	49.2	75.6	16.8	41.8	38.2	35.2	57.9	54.7	61.7	70.8	77.4
CM^r	avg std	**38.5** ±10.6	**41.0** ±7.1	**64.9** ±8.1	9.7 ±0.7	21.2 ±0.7	18.3 ±0.8	**54.8** ±1.2	**63.8** ±0.6	**69.5** ±2.0	**57.3** ±1.7	67.0 ±1.2	**72.0** ±2.5
	max	56.0	55.2	76.4	10.9	22.6	20.0	57.4	65.2	74.3	60.5	69.1	76.4
CM^p	avg std	32.6 ±12.7	39.2 ±8.1	60.2 ±10.0	16.3 ±2.4	28.8 ±2.4	30.8 ±2.5	**55.4** ±3.0	**64.0** ±2.0	**71.0** ±3.8	**57.3** ±2.0	66.9 ±1.3	**72.3** ±2.3
	max	62.9	57.8	80.9	22.0	34.1	36.1	62.4	67.8	78.8	60.1	68.6	75.0

Table 12 Hyper-parameters used for the experiments in Sect. 5.2

	MNIST	fMNIST	USPS	CIFAR10	R10K	20News	10×73k	Pendigit
α	177	80	40	164	10	11	1000	13
B	111	35	150	350	400	85	500	80

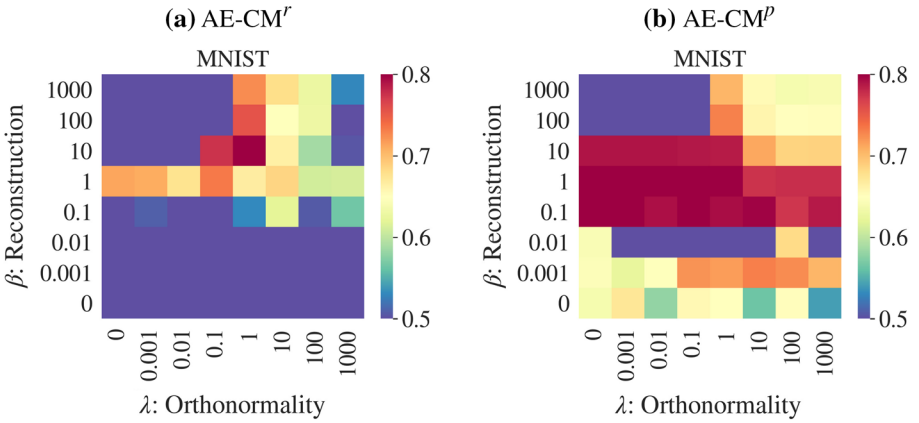

Fig. 12 Clustering performance (ARI) of AE-CM on MNIST for different values of β and λ and initialization scheme

A vanilla CM (Fig. 11b) correctly positions five pairs of centroids on top of the true cluster centroids. Without E_3 (Fig. 11a), the model fails to merge the clusters properly. While five centroids are close to each of the true cluster, the five remaining are gathered around 0. Conversely, if E_3 is weighted stronger (Fig. 11c), the model becomes so prone to merge clusters that it partitions the left cloud using only two groups of centroids.

CM: clustering performance

Table 11 contains the full clustering results for CM, including standard deviation and the best run.

CM: empirical setting

For the experiments reported in Sect. 5.2, the clustering module is trained over 150 epochs using the Adam optimizer (learning rate=0.001). The concentration α and batch-size B used for each dataset are reported in Table 12. The hyper-parameters were optimized using Bayesian optimization over 2000 iterations.

Table 13 Hyper-parameters used to train AE-CM for the experiments in Sect. 5.3

	MNIST	fMNIST	USPS	CIFAR10	R10K	20News	10×73k	Pendigit
α	230	13	20	64	2	10	7	13
β	5	47	0.5	1	1	232	15	0.5
λ	1	1	1	1	1	1	1	1
B	500	175	256	256	256	300	7	100
p	10	10	10	10	100	100	10	10

Table 14 Hyper-parameters used to train the baselines for the experiments in Sect. 5.3

	MNIST	fMNIST	USPS	CIFAR10	R10K	20News	10×73k	Pendigit
u	140	140	30	140	20	20	20	20
γ	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
λ^r	0.1	0.01	0.01	0.01	0.01	10^{-4}	10^{-4}	10^{-4}
λ^p	1.0	0.01	0.1	0.1	1.0	0.01	10^{-4}	10^{-4}
B	256	256	256	256	256	256	256	256

Appendix 3: supplementary materials for AE-CM

AE-CM: hyper-parameters

Besides the architecture of the DAE, AE-CM has two hyper-parameters more than CM: β weighting the reconstruction of the DAE and the Lagrange coefficient λ enforcing the orthonormality of the centroids. To visualize their influence on the clustering performance, a AE-CM is trained on MNIST with various values of β and λ. Each combination is repeated five times. Note that when $\beta = 0$ the setting is equivalent to training only the encoder of the DAE (akin to DEC). Also, if $\lambda = 0$ the orthogonality constraint is omitted. Figure 12 represents the average ARI scores for each combination and both initialization schemes as a heat-map.

With a random initialization (Fig. 12a), if both β and λ are not large enough, the clustering fails, excepted when $\beta = 1$. In that case, the model performs well for every value of λ, even for $\lambda = 0$, i.e., without the orthonormality constraint. Conversely, the AE-CM^r always fails if trained without the reconstruction of the DAE, ($\beta = 0$) . As the order of magnitude of both parameters increases, the performances worsen.

The distribution of AE-CM^p (Fig. 12b) presents similarities with the previous one. Overall the average performances are better for each combination. When β is very small, the ARI exceeds 0.5. The performance also decrease as β and λ become larger. Most noticeable, the band around $\beta = 1$ is still there, but it is thicker. This is in line with the similar analysis on CM (Sect. 1): Pre-trained models are less sensitive to hyperparameters.

Table 15 Clustering performance of AE-CM and its baselines.

Model		MNIST			fMNIST			USPS			CIFAR10		
		ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC
AE+KM	avg std	65.6 ±1.0	71.5 ±0.7	78.6 ±0.6	39.0 ±0.9	55.6 ±0.5	53.0 ±2.3	57.1 ±1.0	64.6 ±0.9	67.5 ±1.0	3.2 ±0.3	6.5 ±0.5	18.9 ±0.5
	max	67.7	73.0	79.6	41.0	56.7	56.9	59.4	66.6	69.5	3.7	7.2	20.2
DCN^r	avg std	10.1 ±11.0	25.6 ±17.4	25.4 ±9.3	17.0 ±11.7	33.5 ±16.9	29.0 ±11.3	17.9 ±11.5	36.5 ±11.1	37.9 ±7.5	3.2 ±2.2	5.9 ±3.5	18.0 ±4.6
	max	30.4	51.6	45.1	40.7	57.5	52.4	44.1	56.0	52.2	5.8	10.3	22.9
DCN^p	avg std	75.6 ±1.4	**82.5** ±1.2	83.1 ±0.9	38.6 ±2.0	57.1 ±0.8	53.1 ±1.8	63.9 ±1.2	73.1 ±1.0	72.5 ±0.7	0.1 ±0.4	0.6 ±1.2	10.7 ±1.5
	max	77.6	85.0	84.5	41.7	58.6	56.4	67.4	75.5	74.5	1.3	4.0	14.5
DEC^r	avg std	11.1 ±2.6	19.0 ±3.2	28.9 ±3.6	22.9 ±3.9	38.1 ±4.1	39.2 ±4.2	36.3 ±4.6	46.9 ±4.5	46.8 ±4.0	3.1 ±0.9	5.7 ±1.5	18.6 ±1.4
	max	16.7	24.1	36.6	29.7	47.0	47.0	43.0	55.0	53.7	4.7	8.6	21.7
DEC^p	avg std	73.8 ±0.7	79.0 ±0.4	83.1 ±0.5	41.9 ±2.0	58.6 ±1.9	54.8 ±2.2	**70.0** ±0.8	**78.1** ±0.6	**76.3** ±0.6	3.1 ±1.4	5.6 ±2.4	18.2 ±3.6
	max	75.2	79.5	84.1	45.9	60.6	58.4	71.3	79.2	77.3	4.5	7.8	21.8
IDEC^r	avg std	27.5 ±5.0	39.0 ±5.2	42.5 ±4.8	35.2 ±5.1	50.8 ±5.6	48.1 ±5.9	41.8 ±8.0	53.2 ±7.0	54.0 ±5.8	2.2 ±2.3	3.6 ±3.6	14.0 ±3.9
	max	38.3	50.1	50.2	48.4	62.5	62.5	53.3	64.4	64.9	5.3	9.3	20.4
IDEC^p	avg std	74.9 ±1.0	80.1 ±0.6	83.4 ±0.6	42.8 ±1.8	59.8 ±0.7	55.4 ±2.2	**70.0** ±0.8	**78.0** ±0.7	**76.1** ±0.6	4.2 ±0.4	7.4 ±0.7	20.2 ±1.1
	max	77.2	81.3	84.9	47.0	61.3	59.2	71.5	79.3	77.0	4.9	8.5	22.0
DKM^r	avg std	72.5 ±5.0	77.3 ±3.1	81.2 ±5.0	41.8 ±2.2	56.4 ±0.9	54.6 ±3.3	58.3 ±4.1	67.0 ±2.4	68.6 ±4.5	**5.8** ±0.7	**9.9** ±1.2	21.3 ±2.1
	max	83.6	83.5	92.3	46.0	58.6	59.4	65.4	70.9	77.1	7.1	12.6	24.3
DKM^p	avg std	74.0 ±2.8	78.3 ±1.7	82.7 ±3.0	36.2 ±3.0	52.0 ±2.3	47.0 ±3.7	60.4 ±3.3	71.8 ±1.9	68.9 ±3.7	**5.9** ±0.4	**10.0** ±0.6	19.7 ±1.2
	max	76.9	80.2	85.3	40.5	56.3	51.5	67.6	76.0	74.9	6.2	10.8	20.8
Cluster GAN^r	avg std	63.6 ±8.0	71.8 ±5.1	76.8 ±6.5	**46.5** ±1.6	**60.7** ±1.3	**59.0** ±1.3	57.4 ±2.2	67.9 ±1.6	70.0 ±2.3	3.2 ±0.5	7.6 ±0.9	20.4 ±0.8
	max	80.3	81.6	90.1	48.9	62.7	61.7	61.4	71.2	73.9	4.4	10.0	21.7
VIB-GMM^r	avg std	73.3 ±7.9	78.3 ±5.1	81.5 ±6.9	43.7 ±3.3	58.4 ±2.3	**59.3** ±4.1	59.9 ±3.5	67.7 ±2.9	68.4 ±4.3	**6.0** ±0.2	**10.1** ±0.2	**24.0** ±0.5
	max	89.9	89.0	95.2	49.9	62.7	67.3	67.5	73.2	79.3	6.5	10.3	24.8
AE-CM^r	avg std	**77.9** ±4.0	80.9 ±2.4	**86.1** ±3.2	43.7 ±2.9	55.6 ±1.8	**59.2** ±3.5	55.1 ±4.5	63.4 ±3.6	65.8 ±4.7	4.1 ±0.8	7.5 ±1.3	20.4 ±1.5
	max	88.6	87.2	94.6	48.9	58.5	65.6	60.6	67.4	72.2	5.3	9.3	22.5
AE-CM^p	avg std	**79.4** ±0.4	**82.4** ±0.4	**86.5** ±0.4	43.1 ±2.6	56.3 ±1.7	**58.5** ±2.8	**69.7** ±4.1	76.7 ±2.3	**76.8** ±3.8	4.1 ±0.6	7.6 ±1.1	20.2 ±0.8
	max	80.3	83.2	87.3	48.4	58.5	64.9	80.3	80.5	87.5	5.6	9.7	21.4

Table 15 (continued)

Model		R10K			20News			10x73k			Pendigit		
AE+KM	avg std max	61.0 ±3.5 67.3	56.8 ±3.1 62.2	74.5 ±3.3 83.3	11.3 ±1.6 13.8	27.4 ±2.5 31.0	24.8 ±2.5 28.6	54.3 ±6.6 64.5	72.5 ±3.0 78.3	64.4 ±4.8 72.4	55.2 ±3.6 62.9	68.2 ±1.7 71.5	70.2 ±4.3 78.5
DCN^r	avg std max	18.0 ±10.3 40.1	19.3 ±8.8 35.1	49.5 ±7.8 63.9	0.0 ±0.1 0.3	0.2 ±0.3 0.9	5.6 ±0.5 6.8	5.3 ±6.9 26.6	17.2 ±16.7 52.3	23.9 ±7.2 38.9	0.1 ±0.4 2.0	0.8 ±2.6 11.3	10.8 ±1.2 15.5
DCN^p	avg std max	**65.4** ±1.9 67.1	**61.1** ±1.0 62.3	**76.6** ±1.1 80.2	11.7 ±2.0 16.1	33.5 ±2.8 37.4	25.3 ±2.8 30.4	9.6 ±22.0 65.8	13.8 ±27.3 80.4	25.2 ±18.6 72.4	56.8 ±4.4 66.1	72.0 ±1.7 75.9	70.8 ±4.7 79.2
DEC^r	avg std max	12.2 ±6.6 26.1	13.2 ±6.3 27.0	43.8 ±6.1 56.2	3.2 ±0.9 4.3	7.8 ±2.2 10.6	10.0 ±1.1 10.7	31.4 ±8.6 53.1	43.5 ±9.6 66.3	46.3 ±8.1 68.4	36.8 ±6.7 49.6	52.3 ±6.9 65.7	49.3 ±6.1 61.7
DEC^p	avg std max	56.8 ±1.9 60.7	56.0 ±2.0 59.6	72.8 ±2.0 77.6	5.5 ±0.6 6.5	11.3 ±1.3 13.7	11.8 ±0.3 12.2	53.5 ±18.6 73.3	67.1 ±22.8 83.4	62.1 ±15.9 78.5	59.6 ±3.5 65.8	72.8 ±1.6 75.7	72.2 ±4.0 79.3
IDEC^r	avg std max	8.6 ±5.6 21.7	9.5 ±4.9 21.8	44.1 ±5.6 55.3	0.0 ±0.0 0.1	0.1 ±0.3 1.1	5.5 ±0.4 6.7	33.7 ±11.6 67.5	46.5 ±12.9 82.8	44.4 ±8.5 69.6	43.3 ±6.9 57.3	61.2 ±5.7 70.7	53.9 ±7.5 69.3
IDEC^p	avg std max	59.7 ±1.3 62.5	56.3 ±0.9 57.8	73.9 ±1.6 78.8	5.9 ±0.4 6.6	12.6 ±1.2 15.0	12.0 ±0.2 12.4	60.1 ±4.9 73.4	75.9 ±4.0 83.5	66.5 ±3.9 78.8	57.9 ±3.8 65.7	71.6 ±1.8 75.1	71.0 ±4.3 79.0
DKM^r	avg std max	51.3 ±4.9 63.3	49.5 ±4.4 58.9	72.3 ±3.3 81.0	4.7 ±0.9 5.6	14.1 ±2.5 17.8	10.9 ±1.1 13.0	65.5 ±5.1 77.0	71.3 ±3.7 78.5	77.0 ±5.0 88.1	52.4 ±3.1 60.3	65.6 ±1.6 68.8	66.9 ±3.3 73.6
DKM^p	avg std max	57.7 ±1.1 59.5	55.5 ±1.3 58.1	**76.5** ±1.8 78.5	20.9 ±6.5 30.7	39.2 ±5.3 46.7	34.3 ±6.9 44.0	38.1 ±14.6 57.2	55.4 ±11.4 69.2	51.6 ±10.2 63.6	15.4 ±11.1 33.1	27.4 ±18.0 50.2	25.1 ±10.7 46.1
Cluster GAN^r	avg std max	33.7 ±11.6 47.8	35.5 ±11.7 50.3	61.4 ±8.6 71.5	18.6 ±2.2 22.0	34.1 ±2.6 38.8	34.1 ±2.3 39.7	39.5 ±2.0 43.5	52.1 ±2.1 55.9	55.5 ±2.6 61.6	**62.9** ±1.7 66.8	74.2 ±1.3 76.5	**75.6** ±2.1 78.0
VIB-GMM^r	avg std max	27.8 ±7.1 43.9	28.7 ±6.3 42.5	56.6 ±4.9 64.9	0.0 ±0.0 0.0	0.0 ±0.0 0.0	0.0 ±0.0 0.0	51.5 ±27.2 81.1	60.7 ±30.7 83.8	60.0 ±23.3 87.0	**64.5** ±4.3 72.3	**75.7** ±2.8 80.1	**74.2** ±3.5 82.7

Table 15 (continued)

Model		R10K			20News			10x73k			Pendigit		
AE-CM^r	avg std max	42.9 ±11.9 62.7	45.6 ±6.9 57.6	67.7 ±7.3 79.6	**31.5** ±4.6 38.7	**45.3** ±3.3 50.8	**43.3** ±4.1 50.5	73.1 ±5.9 85.6	79.0 ±3.7 86.2	80.4 ±5.4 92.1	**64.6** ±4.1 75.3	**75.0** ±2.8 82.1	**75.7** ±4.1 84.3
AE-CM^p	avg std max	64.1 ±2.0 66.7	60.0 ±1.3 62.5	**76.3** ±1.8 82.3	16.8 ±2.5 21.2	29.0 ±2.7 33.5	32.5 ±2.9 37.8	**82.3** ±5.7 86.9	**83.7** ±3.4 86.8	**89.4** ±5.0 92.9	**60.1** ±14.4 69.8	**70.4** ±16.4 78.2	**73.0** ±14.8 81.4

AE-CM: empirical setting

For the experiments reported in Sect. 5.3, AE-CM is trained over 150 epochs using the Adam optimizer (learning rate=0.001). Each layer of the DAE is activated with a leaky -ReLU with a slope of 0.2, except for the last one of the encoder and of the decoder. AE-CM depends on four hyper-parameters: the weight $\beta > 0$, the concentration $\boldsymbol{\alpha} \in \mathbb{S}^K$, the Lagrange multiplier $\lambda > 0$ and the size of the batches $B \in \mathbb{N}^*$. The four hyper-parameters plus the dimension of the feature space, p; were optimized using Bayesian optimization over 2000 iterations. The selected values are reported in Table 13.

The same architecture is used for the baselines, except for DKM where the activation are all ReLU. DEC, IDEC and DCN update their clustering every u iterations, IDEC and DCN rely on a hyper-parameter γ and DKM on a λ. Regarding DKM, the annealing process of the softmax parameter is updated every 5 epochs. We report in Tables 13 and 14 the values used for each dataset.

AE-CM: clustering performance

Table 15 contains the full clustering results for AE-CM, including standard deviation and the best run.

Funding Open access funding provided by UiT The Arctic University of Norway (incl University Hospital of North Norway).

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)* (pp. 265–283).

Alemi, A. A., Fischer, I., Dillon, J. V., & Murphy, K. (2016). Deep variational information bottleneck. http://arxiv.org/abs/1612.00410

Alimoglu, F., & Alpaydin, E. (1996). Methods of combining multiple classifiers based on different representations for Pen-based Handwritten Digit Recognition. In *Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium*.

Arthur, D., & Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (pp. 1027–1035).

Bezdek, J. C. (2013). *Pattern recognition with fuzzy objective function algorithms*. Berlin: Springer.

Bianchi, F. M., Grattarola, D., & Alippi, C. (2020). Spectral clustering with graph neural networks for graph pooling. In *Proceedings of the 37th of the International Conference on Machine Learning (ICML)* (pp. 874–883).

Bishop, C. M. (2006). *Pattern recognition and machine learning (Information science and statistics)*. Berlin, Heidelberg: Springer-Verlag.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. Monterey, CA: Wadsworth and Brooks.

Caron, M., Bojanowski, P., Joulin, A., & Douze, M. (2018). Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 132–149).

Chang, J., Wang, L., Meng, G., Xiang, S., & Pan, C. (2017). Deep adaptive image clustering. In *Proceedings of the IEEE international conference on computer vision (ICCV)* (pp. 5879–5887).

Dhillon, I. S., Guan, Y., & Kulis, B. (2004). Kernel k-means: Spectral clustering and normalized cuts. In *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '04*.

Dilokthanakul, N., Mediano, P. A. M., Garnelo, M., Lee, M. C. H., Salimbeni, H., Arulkumaran, K., & Shanahan, M. (2016). Deep unsupervised clustering with Gaussian mixture variational autoencoders. http://arxiv.org/abs/1611.02648

Dunn, J. C. (1973). A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, *3*(3), 32–57.

Estévez, P. A., Tesmer, M., Perez, C. A., & Zurada, J. M. (2009). Normalized mutual information feature selection. *IEEE Trans. Neural Netw., 20*(2), 189–201.

Fard, M. M., Thonet, T., & Gaussier, E. (2020). Deep k-means: Jointly clustering with k-means and learning representations. *Pattern Recogn. Lett., 138,* 185–192.

Frandsen, P. B., Calcott, B., Mayer, C., & Lanfear, R. (2015). Automatic selection of partitioning schemes for phylogenetic analyses using iterative k-means clustering of site rates. *BMC Evolut. Biol., 15*(1), 1–17.

Gasch, A. P., & Eisen, M. B. (2002). Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biol., 3*(11), 1–22.

Dizaji, K. G., Herandi, A., Deng, C., Cai, W., & Huang, H. (2017). Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE international conference on computer vision (ICCV)* (pp. 5736–5745).

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Networks. http://arxiv.org/abs/1406.2661

Guo, X., Gao, L., Liu, X., & Yin, J. (2017). Improved Deep Embedded Clustering with local structure preservation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 1753–1759).

Guo, X., Liu, X., Zhu, E., & Yin, J. (2017). Deep Clustering with Convolutional Autoencoders. In *Neural Information Processing* (pp. 373–382).

Guyon, I., Boser, B., & Vapnik, V. (1993). Automatic capacity tuning of very large VC-dimension classifiers. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 147–155).

Haeusser, P., Plapp, J., Golkov, V., Aljalbout, E., & Cremers, D. (2018). Associative deep clustering: Training a classification network with no labels. In *German Conference on Pattern Recognition* (pp. 18–32).

Hasnat, M. A., Bohné, J., Milgram, J., Gentric, S., & Chen, L. (2017). von Mises-Fisher mixture model-based deep learning: Application to face verification. http://arxiv.org/abs/1706.04264

Hennig, C., & Liao, T. F. (2013). How to find an appropriate clustering for mixed-type variables with application to socio-economic stratification. *J. Royal Stat. Soc.: Ser. C (Appl. Stat.), 62*(3), 309–369.

Hu, W., Miyato, T., Tokui, S., Matsumoto, E., & Sugiyama, M. (2017). Learning discrete representations via information maximizing self-augmented training. In *Proceedings of the 34th International Conference on Machine Learning (ICML)* (pp. 1558–1567).

Hubert, L., & Arabie, P. (1985). Comparing partitions. *J. Classif., 2*(1), 193–218.

Jang, E., Gu, S., & Poole, B. (2016). Categorical reparameterization with Gumbel-Softmax. http://arxiv.org/abs/1611.01144

Ji, X., Henriques, J. F., & Vedaldi, A. (2019). Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 9865–9874).

Jiang, Z., Zheng, Y., Tan, H., Tang, B., & Zhou, H. (2017). Variational deep embedding: an unsupervised and generative approach to clustering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 1965–1972).

Kampffmeyer, M., Løkse, S., Bianchi, F. M., Livi, L., Salberg, A. B., & Jenssen, R. (2019). Deep divergence-based approach to clustering. *Neural Netw., 113,* 91–101.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. http://arxiv.org/abs/1412.6980

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. http://arxiv.org/abs/1312.6114

Krishna, K., & Murty, M. N. (1999). Genetic k-means algorithm. *IEEE Trans. Syst. Man Cybern. Part B, 29*(3), 433–439.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical Report TR-2009, University of Toronto, Toronto.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Res. Logist. Quarterly, 2*(1–2), 83–97.

Kulis, B., Jordan, M.I. (2012). Revisiting k-means: New algorithms via bayesian nonparametrics. In *Proceedings of the 29th International Conference on Machine Learning (ICML)* (pp. 1131–1138).

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Comput., 1*(4), 541–551.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE Trans. Inform. Theory, 28*(2), 129–137.

Lücke, J., & Forster, D. (2019). k-means as a variational em approximation of gaussian mixture models. *Pattern Recogn. Lett., 125,* 349–356.

Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *J. Mach. Learn. Res., 9,* 2579–2605.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015). Adversarial autoencoders. http://arxiv.org/abs/1511.05644

McConville, R., Santos-Rodriguez, R., Piechocki, R. J., & Craddock, I. (2021). N2D: (not too) deep clustering via clustering the local manifold of an autoencoded embedding. In *2020 25th International Conference on Pattern Recognition (ICPR)* (pp. 5145–5152).

McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform manifold approximation and projection for dimension reduction. http://arxiv.org/abs/1802.03426

Miklautz, L., Mautz, D., Altinigneli, M. C., Böhm, C., & Plant, C. (2020). Deep embedded non-redundant clustering. *Proc. AAAI Conf. Artif. Intell., 34,* 5174–5181.

Mukherjee, S., Asnani, H., Lin, E., & Kannan, S. (2019). Clustergan: Latent space clustering in generative adversarial networks. *Proc. AAAI Conf. Artif. Intell., 33,* 4610–4617.

Punj, G., & Stewart, D. W. (1983). Cluster analysis in marketing research: Review and suggestions for application. *J. Marketing Res., 20*(2), 134–148.

Rosenberg, A., & Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)* (pp. 410–420).

Salaün, A., Petetin, Y., & Desbouvries, F. (2019). Comparing the modeling powers of RNN and HMM. In *2019 18th IEEE International Conference On Machine Learning and Applications (ICMLA)* (pp. 1496–1499).

Springenberg, J. T. (2015). Unsupervised and semi-supervised learning with categorical generative adversarial networks. http://arxiv.org/abs/1511.06390

Tian, F., Gao, B., Cui, Q., Chen, E., & Liu, T. Y. (2014). Learning deep representations for graph clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence* (pp. 1293–1299).

Tishby, N., Pereira, F. C., & Bialek, W. (2000). The information bottleneck method. http://arxiv.org/abs/physics/0004057

Uğur, Y., Arvanitakis, G., & Zaidi, A. (2020). Variational information bottleneck for unsupervised clustering: Deep gaussian mixture embedding. *Entropy, 22*(2), 213.

Van Gansbeke, W., Vandenhende, S., Georgoulis, S., Proesmans, M., Van Gool, L. (2020). Scan: Learning to classify images without labels. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 268–285).

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. http://arxiv.org/abs/1708.07747

Xie, J., Girshick, R., Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)* (pp. 478–487).

Yang, B., Fu, X., Sidiropoulos, N. D., & Hong, M. (2017). Towards K-means-friendly spaces: simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning (ICML)* (pp. 3861–3870).

Yang, J., Parikh, D., Batra, D. (2016). Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 5147–5156).

Yang, L., Cheung, N.M., Li, J., Fang, J. (2019). Deep clustering by gaussian mixture variational autoencoders with graph embedding. In *Proceedings of the IEEE international conference on computer vision (ICCV)* (pp. 6440–6449).

Yang, X., Deng, C., Wei, K., Yan, J., & Liu, W. (2020). Adversarial Learning for Robust Deep Clustering. In *Advances in Neural Information Processing Systems (NeurIPS)* (pp. 9098–9108).

Zhang, T., Ji, P., Harandi, M., Huang, W., Li, H. (2019). Neural collaborative subspace clustering. In *Proceedings of the 36th International Conference on Machine Learning (ICML)* (pp. 7384–7393).

Zheng, G. X., Terry, J. M., Belgrader, P., Ryvkin, P., Bent, Z. W., Wilson, R., et al. (2017). Massively parallel digital transcriptional profiling of single cells. *Nature communications, 8*(1), 1–12.