DeepDPM: Deep Clustering With an Unknown Number of Clusters

Meitar Ronen Shahaf E. Finder Oren Freifeld
The Department of Computer Science, Ben-Gurion University of the Negev
meitarr@post.bgu.ac.il finders@post.bgu.ac.il orenfr@cs.bgu.ac.il

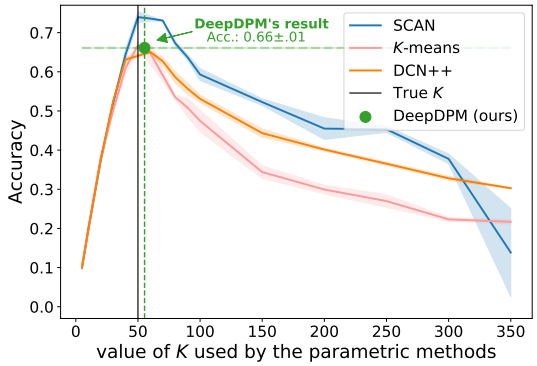
Abstract

*Deep Learning (DL) has shown great promise in the unsupervised task of clustering. That said, while in classical (*i.e.*, non-deep) clustering the benefits of the nonparametric approach are well known, most deep-clustering methods are parametric: namely, they require a predefined and fixed number of clusters, denoted by K. When K is unknown, however, using model-selection criteria to choose its optimal value might become computationally expensive, especially in DL as the training process would have to be repeated numerous times. In this work, we bridge this gap by introducing an effective deep-clustering method that does not require knowing the value of K as it infers it during the learning. Using a split/merge framework, a dynamic architecture that adapts to the changing K, and a novel loss, our proposed method outperforms existing nonparametric methods (both classical and deep ones). While the very few existing deep nonparametric methods lack scalability, we demonstrate ours by being the first to report the performance of such a method on ImageNet. We also demonstrate the importance of inferring K by showing how methods that fix it deteriorate in performance when their assumed K value gets further from the ground-truth one, especially on imbalanced datasets. Our code is available at* https://github.com/BGU-CS-VIL/DeepDPM.

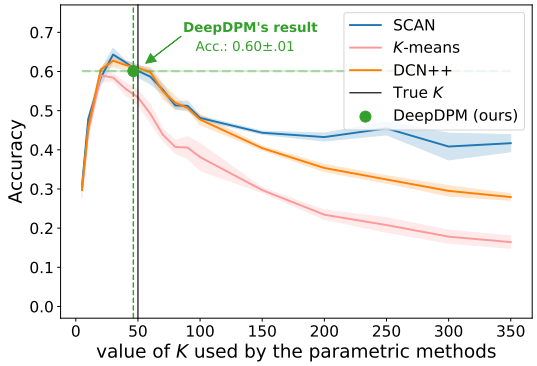
1. Introduction

Clustering is an important unsupervised-learning task where, unlike in the supervised case of classification, class labels are unavailable. Moreover, in the purely-unsupervised (and more realistic) setting this work focuses on, the number of classes, denoted by K, and their relative sizes (*i.e.*, the class weights) are unknown too.

Acknowledgements. This work was supported by the Lynn and William Frankel Center at BGU CS, by the Israeli Council for Higher Education via the BGU Data Science Research Center, and by Israel Science Foundation Personal Grant #360/21. M.R. was also funded by the VATAT National excellence scholarship for female Master's students in Hi-Tech-related fields.



(a) ImageNet50: The original balanced dataset



(b) ImageNet50: An imbalanced dataset

Figure 1. Mean clustering accuracy of 3 runs ($\pm$ std. dev.) on ImageNet50. The Ground Truth K is 50. Parametric methods such as K-means, DCN++ (an improved variant of [71]) and SCAN [64], require knowing K. When given a poor estimate of K, they deteriorate in performance in a balanced dataset (a) and even more so in an imbalanced dataset (b). In contrast, the proposed DeepDPM does not require knowing K (it infers its value; *e.g.*, $K = 55.3 \pm 1.53$ in (a) and 46.3 ± 2.52 in (b)) and yet yields comparable results.

The emergence of Deep Learning (DL) has not skipped clustering tasks. DL methods usually cluster large and high-dimensional datasets better and more efficiently than classical (*i.e.*, non-deep) clustering methods [64, 71]. That said, while in classical clustering it is well understood that *nonparametric* methods (namely, methods that find K) have advantages over *parametric* ones (namely, methods that re-

quire a known K) [8, 57], there are only a few nonparametric deep clustering methods. Unfortunately, the latter are neither scalable nor effective enough. *Our work bridges this gap by proposing an effective deep nonparametric method*, called DeepDPM. In fact, even when K is known, DeepDPM still achieves results comparable to leading parametric methods (especially in imbalanced cases) despite their "unfair" advantage; see, *e.g.*, Figure 1 or § 5.

More generally, the ability to infer the latent K has *practical* benefits, including the following ones. 1) Without a good estimate of K, parametric methods might suffer in performance. Figure 1 shows that using the wrong K can have a significant negative effect on parametric methods in both balanced and imbalanced datasets. When the value of K becomes more and more inaccurate, even a State-Of-The-Art (SOTA) parametric deep clustering method, SCAN [64], deteriorates in performance significantly. 2) Changing K during training has positive optimization-related implications; *e.g.*, by splitting a single cluster into two, multiple data labels are changed *simultaneously*. This often translates to large moves on the optimization surface which may lead to convergence to better local optima and performance gains [10]; *e.g.*, in § 5 we demonstrate cases where nonparametric methods, ours included, outperform parametric ones even when the latter are given the true K. 3) A common workaround to not knowing K is to use *model selection*: namely, run a parametric method numerous times, using different K values over a wide range, and then choose the "best" K via an unsupervised criterion. That approach, however, besides missing the aforementioned potential gains (not being able to make large moves), does not scale and is usually infeasible for large datasets, *especially in DL*. Moreover, *the negative societal impact of the model-selection approach must be noted as well*: training a deep net tens or hundreds of times on a large dataset consumes prohibitively-large amounts of energy. 4) K itself may be a sought-after quantity of importance.

Bayesian nonparametric (BNP) mixture models, exemplified by the Dirichlet Process Mixture (DPM) model, offer an elegant, data-adaptive, and mathematically-principled solution for clustering when K is unknown. However, the high computational cost typically associated with DPM inference is arguably why only a few works tried to use it in conjunction with deep clustering (*e.g.*, [11, 66, 74]). Here we propose to *combine the benefits of DL and the DPM effectively*. The proposed method, DeepDPM, uses splits and merges of clusters to change K together with a dynamic architecture to accommodate for such changes. It also uses a novel amortized inference for Expectation-Maximization (EM) algorithms in mixture models. DeepDPM can be incorporated in deep pipelines that rely on clustering (*e.g.*, for feature learning). Unlike an offline clustering step (*e.g.*, K-means), DeepDPM is differentiable during most of the training (the exception is during the discrete splits/merges) and thus supports gradient propagation through it. DeepDPM outperforms existing nonparametric clustering methods (both classical and deep ones) across several datasets and metrics. It also handles class imbalance gracefully and scales well to large datasets. While we focus on clustering and not feature learning, we also show examples of clustering on pretrained features as well as jointly learning features and clustering in an end-to-end fashion. To summarize, **our key contributions are:** 1) A deep clustering method that infers the number of clusters. 2) A novel loss that enables a new amortized inference in mixture models. 3) A demonstration of the importance, in deep clustering, of inferring K. 4) Our method outperforms existing nonparametric clustering methods and we are the first to report results of a deep nonparametric clustering method on a large dataset such as ImageNet [17].

2. Related Work

Parametric Deep Clustering methods. Recent such works can be divided into two types: two-step approaches and end-to-end ones. In the former, clustering is performed on features extracted in a pretext task. For instance, McConville *et al.* [47] run K-means on the embeddings, transformed by UMAP [48], of a pretrained Autoencoder (AE). While not scalable, [47] achieves competitive results when it is applicable. Another example is SCAN [64] which uses unsupervised pretrained feature extractors (*e.g.*, MoCo [13] and SimCLR [12]). While reaching SOTA results, SCAN, being parametric, depends on having an estimate of K and, as we show, deteriorates in performance when the estimate is too inaccurate. Moreover, SCAN assumes uniform class weights (*i.e.* a balanced dataset) and that is often unrealistic in purely-unsupervised cases.

End-to-end deep methods jointly learn features and clustering, possibly by alternation. Several works use an AE, or a Variational AE (VAE), with an additional clustering loss [40, 68, 70–72]; *e.g.*, DCN [71] runs K-means on the embeddings of a pretrained AE, and retrains it with a loss consisting of a reconstruction term and a clustering-based term, to simultaneously update the features, clusters' centers, and assignments. Other works, *e.g.* [5, 6], use convolutional neural nets to alternately learn features and clustering.

While our work focuses on clustering, not feature learning, we demonstrate how it can also be incorporated with the two approaches above. Moreover, all the methods above assume a predefined and fixed K and, at least the more effective ones among them, take substantial time and resources to train (so searching for the "right" K using model selection is costly and/or inapplicable).

Nonparametric Classical Clustering. Closely related to our work is BNP clustering and, more specifically, the DPM model [1, 24]. While many computer-vision works rely on BNP clustering [4, 9, 14, 25–28, 30, 32, 33, 38, 39, 41, 44–46, 49, 53–59, 62], it has yet to become a mainstream

choice, partly due to the lack of efficient large-scale inference tools. Fortunately, this is starting to change; see, *e.g.*, the highly-effective DPM sampler from [21] (a modern and scalable implementation of the DPM sampler from [10]) or the scalable streaming DPM inference in [20]. Of note, an important alternative to sampling is variational DPM inference [3, 31, 34, 36, 42]. A non-Bayesian example of a popular nonparametric method is DBSCAN [23] which is density-based and groups together closely-packed points. While DBSCAN has efficient implementations, it is highly sensitive to its hyperparameters which are hard to tune.

Nonparametric Deep Clustering. Among the very few examples of deep methods that also find K are [11,52,66,74]. Some of them use an offline DPM inference for pseudo-labels for fine-tuning a deep belief network [11], or an AE [66] (similarly to the parametric methods in [5, 6, 71]). As the methods in [66] and [11] rely on slow DPM samplers, they do not scale to large datasets. AdapVAE [74] uses a DPM prior for a VAE. In DCC [52], feature learning and clustering are performed simultaneously like in [74]; however, instead of ELBO minimization, DCC uses a nearest-neighbor graph to group points that are close in the latent space of an AE. Our method is empirically more effective than [52, 74] and also scales much better. While not a clustering method per-se, and similarly to [47], [65] uses an AE and t-SNE [63] to find K. Like [47], however, [65] does not scale. In [22], a deep net is simultaneously trained on a family of losses instead of a single one. At least in theory, that approach may be adapted to nonparametric clustering but this direction has yet to be explored. Both [60] and [50] do not assume a known K, where the former focuses on clustering faces and the latter on generating posterior samples of cluster labels for any new dataset. Unlike our method, however, [50, 60] are supervised. Similarly, [2] iteratively forms clusters by sequentially examining each sample against the members of existing clusters. The clustering criterion is based on a supervised evaluation net. Lastly, while [73] relies on a BNP mixture, their method (and code) still uses a fixed K.

3. Preliminaries: DPGMM-based Clustering

Let $\mathcal{X} = (\boldsymbol{x}_i)_{i=1}^N$ denote N data points in $\mathbb{R}^d$. The clustering task aims to partition $\mathcal{X}$ into K disjoint groups, where z_i is the point-to-cluster assignment, known as the *cluster label*, of $\boldsymbol{x}_i$. *Cluster* k consists of all the points labeled as k; *i.e.*, $(\boldsymbol{x}_i)_{i:z_i=k}$. The number of clusters, $K \triangleq |\{k : k \in \boldsymbol{z}\}|$, is thus the number of unique elements in $\boldsymbol{z} = (z_i)_{i=1}^N$.

The classical Gaussian Mixture Model (GMM) has a BNP extension: the Dirichlet Process GMM (DPGMM) [1, 24]. Informally, the DPGMM (a specific case of the DPM) entertains the notion of a mixture with infinitely-many Gaussians:

$$p(\boldsymbol{x}|(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k)_{k=1}^\infty) = \sum\nolimits_{k=1}^\infty \pi_k \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (1)$$

where $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is a Gaussian probability density function (pdf) (of mean $\boldsymbol{\mu}_k \in \mathbb{R}^d$ and a d-by-d covariance matrix $\boldsymbol{\Sigma}_k$) evaluated at $\boldsymbol{x} \in \mathbb{R}^d$, $\pi_k > 0\ \forall k$, and $\sum_{k=1}^\infty \pi_k = 1$. For a gentle introduction to the DPGMM with a computer-vision audience in mind, see [8, 57]. Let $\boldsymbol{\theta}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ denote the parameters of Gaussian k. Note the distinction between *component* k (namely, the k-th Gaussian, identified with its parameter, $\boldsymbol{\theta}_k$) and cluster k. The components, $\boldsymbol{\theta} = (\boldsymbol{\theta}_k)_{k=1}^\infty$, and weights, $\boldsymbol{\pi} = (\pi_k)_{k=1}^\infty$, are assumed to be drawn (independently) from their own prior distributions: the weights, $\boldsymbol{\pi}$, are drawn using the Griffiths-Engen-McCloskey stick-breaking process (GEM) [51] with a concentration parameter $\alpha > 0$, while the parameters, $(\boldsymbol{\theta}_k)_{k=1}^\infty$, are independent and identically-distributed (i.i.d.) draws from their prior $p(\boldsymbol{\theta}_k)$, typically a Normal-Inverse Wishart (NIW) distribution. While there are infinitely-many *components*, note that there are still finitely-many *clusters* as the *latent random variable* K is bounded above by N. By possibly renaming cluster indices, we may assume without loss of generality that $\{k : k \in \boldsymbol{z}\} = \{1, 2, \ldots, K\}$.

The DPGMM is often used in clustering when K is unknown. DPGMM inference methods typically seek to find $\boldsymbol{z} = (z_i)_{i=1}^N$ (which implies K) and $(\boldsymbol{\theta}_k, \pi_k)_{k=1}^K$. As explained in our supplementary material (**Supmat**), the inferred value of K is affected by the following factors: $\mathcal{X}$, α, and the NIW hyperparameters. Our method (§ 4) is inspired in part by Chang and Fisher III's DPM sampler [10] which consists of a split/merge framework [37] (which we adopt) and a restricted sampler (which is less relevant to our work).

The split/merge framework augments the latent variables, $(\boldsymbol{\theta}_k)_{k=1}^\infty$, $\boldsymbol{\pi}$, and $(z_i)_{i=1}^N$, with auxiliary variables. To each z_i, an additional *subcluster label*, $\widetilde{z}_i \in \{1, 2\}$, is added. To each $\boldsymbol{\theta}_k$, two subcomponents are added, $\widetilde{\boldsymbol{\theta}}_{k,1}$, $\widetilde{\boldsymbol{\theta}}_{k,2}$, with nonnegative weights $\widetilde{\boldsymbol{\pi}}_k = (\widetilde{\pi}_{k,j})_{j\in\{1,2\}}$ (where $\widetilde{\pi}_{k,1} + \widetilde{\pi}_{k,2} = 1$), forming a 2-component GMM. Next, splits and merges allow changing K via the Metropolis-Hastings framework [29]. That is, during the inference, every certain amount of iterations the split of cluster k into its subclusters is proposed. That split is accepted with probability $\min(1, H_\mathrm{s})$ where

$$H_\mathrm{s} = \frac{\alpha\Gamma(N_{k,1}) f_{\boldsymbol{x}}(\mathcal{X}_{k,1}; \lambda)\Gamma(N_{k,2}) f_{\boldsymbol{x}}(\mathcal{X}_{k,2}; \lambda)}{\Gamma(N_k) f_{\boldsymbol{x}}(\mathcal{X}_k; \lambda)} \quad (2)$$

is the Hastings ratio, Γ is the Gamma function, $\mathcal{X}_k = (\boldsymbol{x}_i)_{i:z_i=k}$ stands for the points in cluster k, $N_k = |\mathcal{X}_k|$, $\mathcal{X}_{k,j} = (\boldsymbol{x}_i)_{i:(z_i,\widetilde{z}_i)=(k,j)}$ denotes the points in subcluster j ($j \in \{1, 2\}$), $N_{k,j} = |\mathcal{X}_{k,j}|$, and $f_{\boldsymbol{x}}(\cdot; \lambda)$ is the *marginal* likelihood where λ represents the NIW hyperparameters. See our **Supmat** for more details. Upon a split proposal acceptance, each of the newly-born clusters is augmented with two subclusters. This ratio, H_s, can be interpreted as comparing the marginal likelihood of the data under the two subclusters with its marginal likelihood under the cluster. Merge proposals are handled similarly (see **Supmat**).

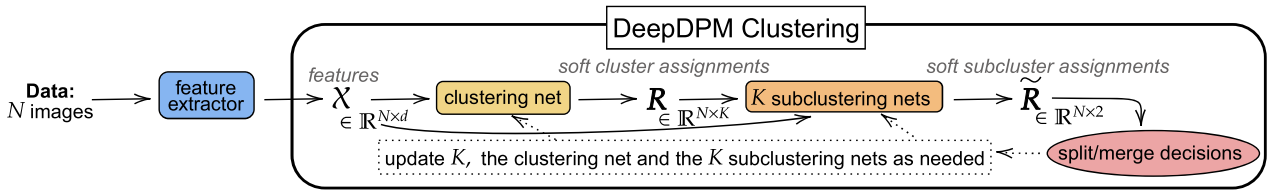


Figure 2. DeepDPM's pipeline: given features $\mathcal{X}$, the clustering net outputs cluster assignments, $\boldsymbol{R}$, while the subclustering nets generate subcluster assignments, $\widetilde{\boldsymbol{R}}$. Upon the acceptance of split/merge proposals, all those nets are updated during the learning.

4. The Proposed Method: DeepDPM

DeepDPM can be viewed as a DPM inference algorithm. Inspired by [10], we use splits and merges to change K where for every cluster we maintain a subcluster pair. For a nominal value of K, rather than resorting to sampling as in [10], we use a deep net trained by a novel amortized inference for EM [16] in a mixture model. DeepDPM has two main parts. The first is a clustering net, while the second consists of K subclustering nets (one for each cluster k, $k \in \{1,\dots,K\}$). In § 4.1 we describe how DeepDPM operates given a nominal value of K and in § 4.2 how K is changed and how our architecture adapts accordingly. We discuss the amortized-inference aspects of our approach in § 4.3, our weak prior in § 4.4, and how DeepDPM may be combined with feature learning in § 4.5. Figure 2 depicts the overall pipeline.

4.1. DeepDPM Under a Fixed K

We start by describing DeepDPM's forward pass during the training. Given a current value of K, the data is first passed to the clustering net, f_{cl}, which generates, for each data point $\boldsymbol{x}_i$, K soft cluster assignments:

$$f_{\text{cl}}(\mathcal{X}) = \boldsymbol{R} = (\boldsymbol{r}_i)_{i=1}^N \qquad \boldsymbol{r}_i = (r_{i,k})_{k=1}^K \tag{3}$$

where $r_{i,k} \in [0,1]$ is the soft assignment of $\boldsymbol{x}_i$ to cluster k (also called the responsibility of cluster k to data point $\boldsymbol{x}_i$) and $\sum_{k=1}^K r_{i,k} = 1$. From $(\boldsymbol{r}_i)_{i=1}^N$ we compute the hard assignments $\boldsymbol{z} = (z_i)_{i=1}^N$ by $z_i = \arg\max_k r_{i,k}$. Next, each subclustering net, f_{sub}^k (where $k \in \{1,\dots,K\}$), is fed with the data (hard-) assigned to its respective cluster (*i.e.*, f_{sub}^k is fed with $\mathcal{X}_k = (\boldsymbol{x}_i)_{i:z_i=k}$) and generates soft subcluster assignments:

$$f_{\text{sub}}^k(\mathcal{X}_k) = \widetilde{\boldsymbol{R}}_k = (\widetilde{\boldsymbol{r}}_i)_{i:z_i=k} \qquad \widetilde{\boldsymbol{r}}_i = (\widetilde{r}_{i,j})_{j=1}^2 \tag{4}$$

where $\widetilde{r}_{i,j} \in [0,1]$ is the soft assignment of $\boldsymbol{x}_i$ to subcluster j ($j \in \{1,2\}$), and $\widetilde{r}_{i,1} + \widetilde{r}_{i,2} = 1 \,\forall k \in \{1,\dots,K\}$. As detailed in § 4.2, the subclusters learned by $(f_{\text{sub}}^k)_{k=1}^K$ are used in split proposals. Each of the $K+1$ nets (f_{cl} and $(f_{\text{sub}}^k)_{k=1}^K$) is a simple multilayer perceptron with a single hidden layer. The last layer of f_{cl} has K neurons while the last layer of each f_{sub}^k has two.

We now introduce a new loss motivated by EM in the Bayesian GMM (though the idea, in fact, also holds in the non-Bayesian GMM case, as well as for EM in parametric mixtures with non-Gaussian components). Concretely, in each epoch, our clustering net is optimized to generate soft assignments that would resemble those obtained by an E step of the EM-GMM algorithm (recall that the E steps of the Bayesian and non-Bayesian EM-GMM coincide). Each E step is followed by a standard M step in a Bayesian GMM, except that the soft assignments used in the Maximum-a-Posterior (MAP) estimates are those produced by our clustering net. We now provide the details. For each $\boldsymbol{x}_i$ and each $k \in \{1,\dots,K\}$ we compute the (standard) E-step probabilities, $\boldsymbol{r}_i^{\text{E}} = (r_{i,k}^{\text{E}})_{k=1}^K$, where

$$r_{i,k}^{\text{E}} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k'=1}^K \pi_{k'} \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})} \qquad k \in \{1,\dots,K\} \tag{5}$$

is computed using $(\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_{k=1}^K$ from the previous epoch. Note that $\sum_{k=1}^K r_{i,k}^{\text{E}} = 1$. We then encourage f_{cl} to generate similar soft assignments using the following new loss:

$$\mathcal{L}_{\text{cl}} = \sum_{i=1}^N \text{KL}(\boldsymbol{r}_i \| \boldsymbol{r}_i^{\text{E}}) \tag{6}$$

where KL is the Kullback-Leibler divergence. Next, after every epoch we perform a Bayesian M step but with a twist. Recall that in this step, one uses the weighted versions of the MAP estimates of $(\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_{k=1}^K$ (computed using standard formulas; see **Supmat**) where the weights are the $r_{i,k}^{\text{E}}$ values (Eq. (5)). We apply the same formulas but instead of $r_{i,k}^{\text{E}}$ we use the $r_{i,k}$ values (*i.e.*, the output of f_{cl}). Note that unlike methods (*e.g.* K-means or SCAN) that enforce/assume uniformity of the weights, our inferred cluster weights, $(\pi_k)_{k=1}^K$, are allowed to deviate from uniformity.

In principle, for $(f_{\text{sub}}^k)_{k=1}^K$ we could have used a loss similar to $\mathcal{L}_{\text{cl}}$. However, here we prefer an isotropic loss:

$$\mathcal{L}_{\text{sub}} = \sum_{k=1}^K \sum_{i=1}^{N_k} \sum_{j=1}^2 \widetilde{r}_{i,j} \|\boldsymbol{x}_i - \widetilde{\boldsymbol{\mu}}_{k,j}\|_{\ell_2}^2 \tag{7}$$

where $N_k = |\mathcal{X}_k|$ and $\widetilde{\boldsymbol{\mu}}_{k,j}$ is the mean of subcluster j of cluster k, computed after every epoch, alongside with subcluster weights and covariances, using weighted MAP estimates similarly to the clusters' case (see **Supmat**). This loss is more efficient than the KL loss while the latter (only in the subcluster case) did not yield improvement. The

iterative process described above needs to be initialized. We do this using K-means (using some initial value of K for the clustering and $K = 2$ for the subclustering). *DeepDPM is fairly robust to the initial K so the latter can be chosen arbitrarily* (see, *e.g.*, § 5.2).

4.2. Changing K via Splits and Merges

During training, we use splits and merges to change K (as in [10]). Every few epochs, we propose either splits or merges. Since K is changing, the architecture, and more specifically, the last layer of the clustering net and the number of subclustering nets, must change too. Of note, the splits/merges facilitate not only changing the value of K but also *large moves*, escaping many poor local optima [10].

Splits. In every split step, we propose to split each of the clusters into its two subclusters. A split proposal is accepted stochastically (as in [10]) with probability $\min(1, H_{\mathrm{s}})$; see Eq. (2). To accommodate for the increase in K, if a split proposal is accepted for cluster k, the k-th unit in the last layer of the clustering net, together with the weights connecting it to the previous hidden layer, is duplicated, and we initialize the parameters of the two new clusters using the parameters learned via the subcluster nets:

$$\begin{aligned} \boldsymbol{\mu}_{k_1} &\leftarrow \widetilde{\boldsymbol{\mu}}_{k,1}\,, \quad \boldsymbol{\Sigma}_{k_1} \leftarrow \widetilde{\boldsymbol{\Sigma}}_{k,1}\,, \quad \pi_{k_1} \leftarrow \pi_k \times \widetilde{\boldsymbol{\pi}}_{k,1} \\ \boldsymbol{\mu}_{k_2} &\leftarrow \widetilde{\boldsymbol{\mu}}_{k,2}\,, \quad \boldsymbol{\Sigma}_{k_2} \leftarrow \widetilde{\boldsymbol{\Sigma}}_{k,2}\,, \quad \pi_{k_2} \leftarrow \pi_k \times \widetilde{\boldsymbol{\pi}}_{k,2} \end{aligned} \tag{8}$$

where k_1 and k_2 denote the indices of the new clusters. We then also add, to each new cluster, a new subclustering net (dynamically allocating the memory).

Merges. When considering merges we must ensure we never simultaneously accept the proposals of, *e.g.*, merging clusters k_1 and k_2 and the merging of clusters k_2 and k_3, thereby mistakenly merging *three* clusters together. Thus, unlike split proposals which are done in parallel, not all possible merges can be considered simultaneously. To avoid sequentially considering all possible merges, we consider (sequentially) the merges of each cluster with only its 3 nearest neighbors. Merge proposals are accepted/rejected using a Hastings ratio, $H_{\mathrm{m}} = 1/H_{\mathrm{s}}$ (as in [10]). If a proposal is accepted, the two clusters are merged and a new subclustering network is initialized. Technically, one of the merged clusters' last layer's units, together with the net's weights connecting it to the previous hidden layer, is removed from f_{cl}, and the parameters and the weight of the newly-born cluster are initialized using the weighted MAP estimates.

4.3. Amortized EM Inference

Suppose we turn off splits/merges and use the Ground Truth (GT) K. Seemingly, this reduces each training epoch to mimicking a single EM iteration. Remarkably, however, and as shown in § 5, even then our method still yields results that are *usually better* than the standard EM. We hypothesize that this stems from the fact that we *amortize* the EM inference; by the virtue of the smoothness of the function learned by the deep net, we improve the prediction for the points in not only the current batch but also other batches. Moreover, the smoothness also serves as an inductive bias such that points which are close in the observation space should have similar labels.

In principle, instead of using our variational loss we could have also used the GMM negative log likelihood (or log posterior). However, empirically that led to unstable optimization and/or poor results. Moreover, basing our loss on matching soft labels rather than likelihood/posterior elegantly makes the method more general: f_{cl} and $\mathcal{L}_{\mathrm{cl}}$ can be used as they are for any component type, not just Gaussians.

4.4. A Weak Prior: Letting the Data Speak for Itself

Recall that the inferred K depends on $\mathcal{X}$, α, and the NIW hyperparameters. We intentionally choose the prior to be *very* weak. Meaning, we choose α as well as the so-called pseudocounts (two of the NIW hyperparameters) to be very low numbers, dwarfed by N, the number of points (see **Supmat** for details). Thus, we let the data, $\mathcal{X}$, to be the most dominant factor in determining K. The weak prior also means that the Bayesian EM-GMM nearly coincides with the non-Bayesian EM-GMM but still helps in the presence of a degenerate sample covariance or very small clusters.

4.5. Feature extraction

To show the effectiveness of our clustering method, we used two types of feature-extraction paradigms: an end-to-end approach, where features and clustering are learned jointly (using alternate optimization), and a two-step approach in which features are learned once (before the clustering) and then held fixed. For the two-step approach, we follow SCAN [64] and use MoCo [13] for (unsupervised) feature extraction. For more details, as well as the scheme we use for an end-to-end feature extraction, see **Supmat**.

5. Results

In this section we evaluate DeepDPM and compare it with several key methods on popular image and text datasets at varying scales. In our evaluations we use three common metrics: clustering accuracy (ACC); Normalized Mutual Information (NMI); Adjusted Rand Index (ARI). The higher the better in all three, and they can accommodate for differences between the inferred K and its GT value. See **Supmat** for more details on the experimental setup and the values of the hyperparameters that we used. Due to space limits, we omit here the NMI and ARI values in several comparisons but these appear in the **Supmat**. We round the results to 2 decimal places so, *e.g.*, a standard deviation (std. dev.) of .00 may still represent a positive (albeit small) number.

Comparing with Classical Methods. We compared DeepDPM with classical parametric methods (K-means;

	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC
	MNIST [18]			USPS [35]			Fashion-MNIST [69]		
K-means^p	.90± .02	.84± .05	.85±.06	.86±.01	.79±.05	.80±.06	.67±.01	.50±.03	.60±.04
GMM^p	**.94±.00**	**.95±.00**	**.98±.00**	.86±.02	.79±.05	.81±.06	.66±.01	.49±.02	.58±.03
DBSCAN	.92±0	.86±0	.89±0	.72±0	.46±0	.57±0	.63±0	-.32±0	.39±0
DPM Sampler	.92±.01	.91±.04	.93±.05	.87±.01	.82±.02	.83±.03	.67±.01	.49± .02	.59±.03
moVB	.93±.00	.94±.00	.97±.00	.87±.02	**.86±.04**	**.90±.04**	.66±.02	.47±.03	.55±.03
DeepDPM (Ours)	**.94±.00**	**.95±.00**	**.98±.00**	**.88±.00**	**.86±.01**	.89±.2	**.68±.01**	**.51±.02**	**.62±.03**
	MNIST^{imb}			USPS^{imb}			Fashion-MNIST^{imb}		
K-means^p	.89± .03	.84± .06	.83±.06	.82±.02	.71±.05	.71±.05	.62±.01	.46±.02	.56±.03
GMM^p	.94±.02	.95±.03	.96±.04	.83±.01	.74±.05	.76±.05	.62±.01	.46±.02	.57±.03
DBSCAN	.93±0	.92±0	.94±0	.84±0	.79±0	.80±0	.62±0	.35±0	.46±0
DPM Sampler	.93±.01	.94±.02	.96±.02	.89±.02	.89±.06	.91±.04	**.66±.01**	**.50± .01**	**.61±.01**
moVB	.94±.00	.95±.00	.96±.00	.88±.01	.89±.02	.91±.02	.63±.01	.44±.02	.53±.02
DeepDPM (Ours)	**.95±.01**	**.97±.01**	**.98±.01**	**.90±.00**	**.92±.00**	**.94±.00**	.65±.00	**.50±.00**	**.61±.00**

Table 1. Comparing the mean results (±std. dev.) of DeepDPM with classical clustering methods. The results are the mean of 10 independent runs. Methods marked with ^{p} are parametric (require K). Datasets marked with ^{imb} are imbalanced ones.

Method	Inferred K		
	MNIST	USPS	Fashion-MNIST
DBSCAN	9.0±0.00	6.0±0.00	4.0±0.00
DPM Sampler	11.3±0.82	8.5±0.85	12.4±0.97
moVB	14±1.00	11.2±1.08	16.9±2.30
DeepDPM (Ours)	**10±0.00**	**9.2±0.42**	**10.2±0.79**

Table 2. Comparing the mean inferred value (±std. dev.) for K of 10 runs among nonparametric methods. GT $K = 10$.

GMM) and nonparametric ones (DBSCAN [23], moVB [34]; the SOTA DPM sampler from [21]). For feature extraction, we performed the process suggested in [47]. We performed the evaluation on the MNIST, USPS, and Fashion-MNIST datasets, as well their imbalanced versions (the latter are defined in the **Supmat**). All the methods used the same (and fixed) data embeddings as input, *and the parametric ones were given the GT K, given them an unfair advantage*. Table 1 shows that DeepDPM almost uniformly dominates across all datasets and metrics, and its performance gain only increases in the imbalanced cases. It is also observable that, compared with the parametric methods, the nonparametric ones (ours included) are less affected by the imbalance. Moreover, Table 2 shows that among the nonparametric methods, DeepDPM's inferred K is the closest to the GT K (see **Supmat** for similar results in the imbalanced case).

Comparing with Deep Nonparametric Methods. As there exist very few deep nonparametric methods, and some of them reported results only on *extremely-small* toy datasets [11, 66] (*e.g.*, one of them stated they could not process even MNIST's train dataset as it was too large for them), we compared DeepDPM with DCC [52] and AdapVAE [74], the only unsupervised deep nonparametric methods that can at least handle the MNIST [18], USPS [35], and STL-10 [15] datasets. As both those methods jointly learn features and clustering, and to show the flexible nature of DeepDPM, we demonstrate its integration with two feature-extraction techniques (described in § 4.5): an end-to-end pipeline (for MNIST and REUTERS-10k [43]) and a two-step approach using features pretrained by MoCo [13] (for STL-10). Unfortunately, we could not run AdapVAE's published code, and thus resort to including the results reported by them. For DCC, using their code we could reproduce their results only on MNIST, so we compare with both the results we managed to obtain using their code and the ones reported by them. Due to these reproducibility issues, we could compare with those methods only on the original (*i.e.*, balanced) datasets. Table 3 shows that DeepDPM outperforms both DCC and AdapVAE. Note we could not find other unsupervised deep nonparametric methods (let alone with available code) that scale to even these fairly-small datasets.

Clustering the Entire ImageNet Dataset. On ImageNet, we obtained the following results: ACC: 0.25, NMI: 0.65, ARI: 0.14. Our method was initialized with $K = 200$ and converged into 707 clusters (GT=1000). These are first results on ImageNet reported for deep nonparametric clustering. Figure 3 shows examples of images clustered together.

5.1. The Value of Deep Nonparametric Methods

When Parametric Methods Break. We study the effect of not knowing K on parametric methods, with and without

	MNIST [18]			STL-10 [15]			Reuters10k [43]		
Method	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC
AdapVAE† [74] *avg*	.86±1.02	.84±2.35	N/A	.75±0.53	**.71±0.81**	N/A	.45±1.79	.43±5.73	N/A
DCC† [52] best	.912	N/A	.96	N/A	N/A	N/A	.59	N/A	.60
DCC‡ [52] *avg*	.90±.02	.89±.07	.91±.07	.22±.00	.01±.00	.04±.00	.25±.00	.00±.00	.00±.00
DeepDPM (ours) *avg*	.90±.01	.91±.02	.93±.03	.78±.004	.70±.01	.84±.01	.61±.00	.64±.01	.83±.00
DeepDPM (ours) best	**.92**	**.93**	**.96**	**.79**	**.71**	**.85**	**.61**	**.64**	**.83**

Table 3. Comparing deep nonparametric methods. †: reported in the papers. ‡: obtained using their code. *avg*: mean (±std. dev.) of 5 runs.

Figure 3. Examples of ImageNet images clustered together by DeepDPM. Each panel stands for a different cluster.

Method	NMI	ARI	ACC
	ImageNet-50: Balanced		
DBSCAN	.52±.00	.09±.00	.24±.00
moVB	.70±.01	.38±.01	.55±.02
DPM Sampler	.72±.00	.43±.01	.57±.01
DeepDPM (ours)	.75±.00	.49±.01	.64±.00
DeepDPM (ours)∗	**.77±.00**	**.54±.01**	**.66±.01**
	ImageNet-50: Imbalanced		
DBSCAN	.33±.00	.04±.00	.24±.00
moVB	.68±.01	.44±.03	.52±.03
DPM Sampler	.70±.00	.40±.01	.51±.00
DeepDPM (ours)	.74±.01	.48±.02	.58±.01
DeepDPM (ours)∗	**.75±.00**	**.51±.01**	**.60±.01**

Table 4. Comparison of nonparametric methods on ImageNet-50 and its imbalanced version. ∗ marks results with AE alternation.

Method	Final/best K: balanced	Final/best K: imbalanced
K-means^p	40	20
DCN++^{p}	60	40
SCAN^p	70	40
DBSCAN	16	13
moVB	46.2±1.3	**46.4±1.1**
DPM Sampler	72.0±2.6	70.3±4.6
DeepDPM (ours)	**52.0±1.0**	43.67±1.2
DeepDPM (ours)∗	55.3±1.5	46.3±2.5

Table 5. Comparing the mean (±std. dev.) value for K found on ImageNet-50 of 3 runs. For the parametric methods (marked with ^{p}) we use the K value with the best silhouette score. ∗ marks results obtained with AE alternation.

class imbalance. We evaluate each method with a wide range of different K values on ImageNet-50. The latter, curated in [64], consists of 50 randomly-selected classes of ImageNet [17]. To generate an imbalanced version of it, we sampled a normalized *nonuniform* histogram from a uniform distribution over the 50-dimensional probability simplex (*i.e.*, all histograms were equally probable) and then sampled examples from the 50 classes in proportions according to that nonuniform histogram. We compared with 3 parametric methods: 1) K-means; 2) the SOTA method SCAN [64]; 3) an improved version of DCN [71], self-coined DCN++, where instead of training an AE on the raw data, we trained it on top of the embeddings SCAN uses (MoCo [13]) where, following [64], we froze those embeddings during training. For DeepDPM, we used the same features.

Since SCAN requires large amounts of memory (*e.g.*, we could only run it on 2 RTX-3090 GPU cards with 24GB memory each, compared with DeepDPM for which a single RTX-2080 (or even GTX-1080) with 8GB sufficed), and due to resource constraints, we were limited in how many K values we could run SCAN with and in the number of times each experiment could run (this high computational cost is one of the problems with model selection in parametric methods). Thus, we collected the results of the parametric methods with K values ranging from 5 to 350. For both the balanced and imbalanced cases, we initialized DeepDPM with $K = 10$. Figure 1 summarizes the ACC results (see **Supmat** for ARI/NMI). As the K value used by the parametric methods diverges from the GT (*i.e.*, $K = 50$), their results deteriorate. Unsurprisingly, when using the GT K, or sufficiently close to it, the parametric methods outperform our

	ACC		
	K_{init}=3	K_{init}=10	K_{init}=30
No splits/merges	.29±.01	.59±.03	.46±.01
No splits	.29±.01	.59±.02	.45±.03
No merges	.46±.00	.58±.01	.47±.01
2-means instead of f_{sub}	.61±.00	.59±.02	.56±.02
No priors in the M step	.58±.01	.57±.02	.58±.01
Isotropic loss instead of $\mathcal{L}_{\text{cl}}$	.58±.00	.58±.00	.58±.02
DeepDPM (full method)	**.62±.03**	**.61±.00**	**.62±.01**

Table 6. DeepDPM's performance under different ablations.

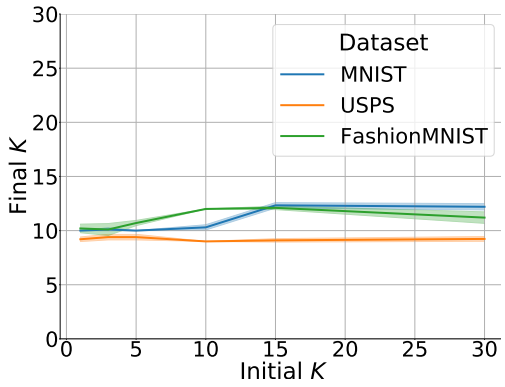


Figure 4. Robustness to the initial K. GT $K = 10$ in all datasets.

nonparametric one, confirming our claim that having a good estimate of K is important for good clustering. Figure 1a, however, shows that even with fairly-moderate deviates from the GT K, DeepDPM's result ($0.66 \pm .01$) surpasses the leading parametric method. Moreover, Figure 1 shows that the parametric SCAN is sensitive to class imbalance; *e.g.*, in Figure 1b, SCAN performs best when $K = 30$ suggesting it is due to ignoring many small classes. In contrast, DeepDPM (scoring $0.60 \pm .01$) is fairly robust to these changes and is comparable in results to SCAN when the latter was given the GT K. In addition, we also show in Table 4 the performance of other nonparametric methods (3 runs on the same features as ours: MoCo+AE). We include DeepDPM's results with alternation (between clustering and feature learning) and without (*i.e.*, holding the features frozen and training DeepDPM only once). Table 5 compares the K values found by the nonparametric methods. DeepDPM inferred a K value close to the GT in both the balanced and imbalanced cases. In the imbalanced case, moVB scored a *slightly* better K but its results (see Table 4) were worse. For the parametric methods, Table 5 also shows the K value of the best silhouette score. The *unsupervised* silhouette metric is commonly used for model selection (NMI/ACC/ARI are supervised, hence inapplicable for model selection). As Table 5 shows, DeepDPM yielded a more accurate K than that approach.

Running Times. Our running time is comparable with a *single* run of SCAN (the SOTA deep parametric method); *e.g.*, on ImageNet-50, SCAN (with 2 NVIDIA 3090 GPUs) trains for $\sim$8 [hr] while ours (with 1 weaker NVIDIA 2080 GPU) takes $\sim$11 [hr]. However, **training SCAN multiple times with a different K each time (as needed for model selection) took more than 3 days.** Thus, DeepDPM's value and positive environmental impact are clear.

5.2. Ablation Study and Robustness to the Initial K

Table 6 quantifies the performance gains due to the different parts of DeepDPM through an ablation study done on Fashion-MNIST (in the setting described earlier). It shows the effect of disabling splits, merges and both; *e.g.*, merges help even when initializing with $K = 3$. In fact, the large moves made by splits/merges help even when $K_{\text{init}} = 10$. Also, replacing the subclustering nets with K-means (using $K = 2$) results in deterioration. Likewise, either turning off the priors when computing the cluster parameters, or using an isotropic loss instead of $\mathcal{L}_{\text{cl}}$, hurts performance and (while not shown here) often destabilizes the optimization. Finally, Figure 4 demonstrates, on three different datasets, DeepDPM's robustness to the initial K.

6. Conclusion

Limitations. As with most clustering methods, if DeepDPM's input features are poor it would struggle to recover. Also, if K is known and the dataset is balanced, parametric methods (*e.g.*, SCAN) may be a slightly better choice.

Future work. An interesting direction is adapting DeepDPM to streaming data (*e.g.*, similarly to how [20] handled streaming DPM inference) or hierarchical settings [7, 19, 61]. Moreover, our results may improve given a more sophisticated framework for building split proposals (*e.g.*, see [67]).

Broader impact. We hope our work will inspire the deep-clustering community to adopt the nonparametric approach as well as raise awareness to issues with the parametric one. Nonparametrics also has an environmental positive impact: obviating the need to repeatedly train deep parametric methods for model selection drastically reduces resource usage.

Summary. We presented a deep nonparametric clustering method, a dynamic architecture that adapts to the varying K values, and a novel loss based on new amortized inference in mixture models. Our method outperforms deep and non-deep nonparametric methods and achieves SOTA results. We demonstrated the issues with parametric clustering, especially the sensitivity to the assumed K, and the added value the nonparametric approach brings to deep clustering. We showed the robustness of our method to both class imbalance and the initial K. Finally, we demonstrated the scalability of DeepDPM by being the first method of its kind to report results on ImageNet. Our code is publicly available.

References

[1] Charles E Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *Annals of Statistics*, 1974. 2, 3

[2] Christos Avgerinos, Vassilios Solachidis, Nicholas Vretos, and Petros Daras. Non-parametric clustering using deep neural networks. *IEEE Access*, 2020. 3

[3] David M Blei and Michael I Jordan. Variational inference for Dirichlet process mixtures. *Bayesian analysis*, 2006. 3

[4] Randi Cabezas, Julian Straub, and John W Fisher III. Semantically-aware aerial reconstruction from multi-modal data. In *ICCV*, 2015. 2

[5] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018. 2, 3

[6] Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Unsupervised pre-training of image features on non-curated data. In *ICCV*, 2019. 2, 3

[7] Chang and Fisher III. Parallel sampling of HDPs using sub-cluster splits. In *NeurIPS*, 2014. 8

[8] Jason Chang. *Sampling in computer vision and Bayesian nonparametric mixtures*. PhD thesis, Massachusetts Institute of Technology, 2014. 2, 3

[9] Jason Chang, Randi Cabezas, and John W Fisher III. Bayesian nonparametric intrinsic image decomposition. In *ECCV*, 2014. 2

[10] Jason Chang and John W Fisher III. Parallel sampling of DP mixture models using sub-cluster splits. In *NeurIPS*, 2013. 2, 3, 4, 5

[11] Gang Chen. Deep learning with nonparametric clustering. *arXiv:1501.03084*, 2015. 2, 3, 6

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 2

[13] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv:2003.04297*, 2020. 2, 5, 6, 7

[14] Anoop Cherian, Vassilios Morellas, Nikolaos Papanikolopoulos, and Saad J Bedros. Dirichlet process mixture models on symmetric positive definite matrices for appearance clustering in video surveillance applications. In *CVPR*, 2011. 2

[15] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *AISTATS*, 2011. 6, 7

[16] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 1977. 4

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2, 7

[18] Li Deng. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 2012. 6, 7

[19] Or Dinari and Oren Freifeld. Scalable and flexible clustering of grouped data via parallel and distributed sampling in versatile hierarchical Dirichlet processes. In *UAI*, 2020. 8

[20] Or Dinari and Oren Freifeld. Sampling in Dirichlet process mixture models for clustering streaming data. In *AISTATS*, 2022. 3, 8

[21] Or Dinari, Angel Yu, Oren Freifeld, and John Fisher III. Distributed MCMC inference in Dirichlet process mixture models using Julia. In *IEEE CCGRID Workshop on High Performance Machine Learning*, 2019. 3, 6

[22] Alexey Dosovitskiy and Josip Djolonga. You only train once: Loss-conditional training of deep networks. In *ICLR*, 2020. 3

[23] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, 1996. 3, 6

[24] Thomas S Ferguson. A Bayesian analysis of some nonparametric problems. *Annals of statistics*, 1973. 2, 3

[25] Soumya Ghosh, Matthew Loper, Erik B Sudderth, and Michael J Black. From deformations to parts: Motion-based segmentation of 3D objects. In *NIPS*, 2012. 2

[26] Soumya Ghosh, Michalis Raptis, Leonid Sigal, and Erik B Sudderth. Nonparametric clustering with distance dependent hierarchies. In *UAI*, 2014. 2

[27] Soumya Ghosh and Erik B Sudderth. Nonparametric learning for layered segmentation of natural images. In *CVPR*, 2012. 2

[28] Ryan Gomes, Max Welling, and Pietro Perona. Incremental learning of nonparametric Bayesian mixture models. In *CVPR*, 2008. 2

[29] Keith W Hastings. Monte Carlo sampling methods using Markov chains and their applications. 1970. 3

[30] David S Hayden, Jason Pacheco, and John W Fisher III. Nonparametric object and parts modeling with Lie group dynamics. In *CVPR*, 2020. 2

[31] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *JMLR*, 2013. 3

[32] Weiming Hu, Xi Li, Guodong Tian, Stephen Maybank, and Zhongfei Zhang. An incremental DPMM-based method for trajectory clustering, modeling, and retrieval. *IEEE TPAMI*, 2013. 2

[33] Michael C Hughes and Erik B Sudderth. Nonparametric discovery of activity patterns from video collections. In *CVPR Workshops*, 2012. 2

[34] Michael C. Hughes and Erik B Sudderth. Memoized online variational inference for Dirichlet process mixture models. In *NeurIPS*, 2013. 3, 6

[35] Jonathan J Hull. A database for handwritten text recognition research. *IEEE TPAMI*, 1994. 6

[36] Viet Huynh, Dinh Phung, and Svetha Venkatesh. Streaming variational inference for Dirichlet process mixtures. In *ACML*, 2016. 3

[37] Sonia Jain and Radford M. Neal. A split-merge Markov chain Monte Carlo procedure for the Dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 2004. 3

[38] Geng Ji, Michael C Hughes, and Erik B Sudderth. From patches to images: a nonparametric generative model. In *ICML*, 2017. 2

[39] Yong-Dian Jian and Chu-Song Chen. Two-view motion segmentation by mixtures of Dirichlet process with model selection and outlier removal. In *ICCV*, 2007. 2

[40] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: A generative approach to clustering. *CoRR*, 2016. 2

[41] Jyri J Kivinen, Erik B Sudderth, and Michael I Jordan. Learning multiscale representations of natural scenes using Dirichlet processes. In *ICCV*, 2007. 2

[42] Kenichi Kurihara, Max Welling, and Nikos Vlassis. Accelerated variational Dirichlet process mixtures. *NeurIPS*, 2007. 3

[43] David D Lewis, Yiming Yang, Tony Russell-Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *JMLR*, 2004. 6, 7

[44] Li-Jia Li and Li Fei-Fei. Optimol: automatic online picture collection via incremental model learning. *IJCV*, 2010. 2

[45] Wenhan Luo, Björn Stenger, Xiaowei Zhao, and Tae-Kyun Kim. Trajectories as topics: Multi-object tracking by topic discovery. *IEEE TIP*, 2018. 2

[46] Amir Markovitz, Gilad Sharir, Itamar Friedman, Lihi Zelnik-Manor, and Shai Avidan. Graph embedded pose clustering for anomaly detection. In *CVPR*, 2020. 2

[47] Ryan McConville, Raul Santos-Rodriguez, Robert J Piechocki, and Ian Craddock. N2d:(not too) deep clustering via clustering the local manifold of an autoencoded embedding. In *ICPR*, 2020. 2, 3, 6

[48] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv:1802.03426*, 2018. 2

[49] Willie Neiswanger, Frank Wood, and Eric Xing. The dependent Dirichlet process mixture of objects for detection-free tracking and object modeling. In *AISTATS*, 2014. 2

[50] Ari Pakman, Yueqi Wang, Catalin Mitelut, JinHyung Lee, and Liam Paninski. Neural clustering processes. In *ICML*. PMLR, 2020. 3

[51] Jim Pitman. Combinatorial stochastic processes. Technical report, Technical Report 621, Dept. Statistics, UC Berkeley, Lecture notes, 2002. 3

[52] Sohil Atul Shah and Vladlen Koltun. Deep continuous clustering. *arXiv:1803.01449*, 2018. 3, 6, 7

[53] Daniel M Steinberg, Oscar Pizarro, and Stefan B Williams. Hierarchical Bayesian models for unsupervised scene understanding. *CVIU*, 2015. 2

[54] Julian Straub, Trevor Campbell, Jonathan P How, and John W Fisher III. Small-variance nonparametric clustering on the hypersphere. In *CVPR*, 2015. 2

[55] Julian Straub, Trevor Campbell, Jonathan P How, and John W Fisher III. Efficient global point cloud alignment using Bayesian nonparametric mixtures. In *CVPR*, 2017. 2

[56] Julian Straub, Oren Freifeld, Guy Rosman, John J. Leonard, and John W. Fisher III. The Manhattan frame model: Manhattan world inference in the space of surface normals. In *IEEE TPAMI*, 2018. 2

[57] Erik B Sudderth. *Graphical models for visual object recognition and tracking*. PhD thesis, Massachusetts Institute of Technology, 2006. 2, 3

[58] Erik B Sudderth and Michael Jordan. Shared segmentation of natural scenes using dependent Pitman-Yor processes. *NIPS*, 2008. 2

[59] Erik B Sudderth, Antonio Torralba, William T Freeman, and Alan S Willsky. Describing visual scenes using transformed objects and parts. *IJCV*, 2008. 2

[60] Makarand Tapaswi, Marc Law, and Sanja Fidler. Video face clustering with unknown number of clusters. In *ICCV*, 2019. 3

[61] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 2006. 8

[62] Roy Uziel, Meitar Ronen, and Oren Freifeld. Bayesian adaptive superpixel segmentation. In *ICCV*, 2019. 2

[63] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *JMLR*, 2008. 3

[64] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. SCAN: Learning to classify images without labels. In *ECCV*, 2020. 1, 2, 5, 7

[65] Yiqi Wang, Zhan Shi, Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep embedding for determining the number of clusters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 3

[66] Zeya Wang, Yang Ni, Baoyu Jing, Deqing Wang, Hao Zhang, and Eric Xing. DNB: A joint learning framework for deep Bayesian nonparametric clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. 2, 3, 6

[67] Vlad Winter, Or Dinari, and Oren Freifeld. Common failure modes of subcluster-based sampling in Dirichlet process gaussian mixture models – and a deep-learning solution. In *AISTATS*, 2022. 8

[68] Lirong Wu, Zicheng Liu, Zelin Zang, Jun Xia, Siyuan Li, Stan Li, et al. Deep clustering and representation learning that preserves geometric structures. *arXiv:2009.09590*, 2020. 2

[69] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. 08 2017. 6

[70] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, 2016. 2

[71] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *ICML*, 2017. 1, 2, 3, 7

[72] Linxiao Yang, Ngai-Man Cheung, Jiaying Li, and Jun Fang. Deep clustering by Gaussian mixture variational autoencoders with graph embedding. In *ICCV*, 2019. 2

[73] Xi Yang, Yuyao Yan, Kaizhu Huang, and Rui Zhang. Vsb-dvm: an end-to-end bayesian nonparametric generalization of deep variational mixture model. In *ICDM*, 2019. 3

[74] Tingting Zhao, Zifeng Wang, Aria Masoomi, and Jennifer G Dy. Streaming adaptive nonparametric variational autoencoder. *arXiv:1906.03288*, 2019. 2, 3, 6, 7

DeepDPM: Deep Clustering With an Unknown Number of Clusters

Supplemental Material

Meitar Ronen Shahaf E. Finder Oren Freifeld
The Department of Computer Science, Ben-Gurion University of the Negev
meitarr@post.bgu.ac.il finders@post.bgu.ac.il orenfr@cs.bgu.ac.il

This document contains additional results, implementation details, and explanations that were omitted from the paper due to page limits.

Contents

1. Additional Results

1.1. Additional Visual Examples of Images Clustered Together

Here, in Figures 1 to 8, we provide additional examples of images from the validation set of the ImageNet dataset that were clustered together using DeepDPM. These figures show how DeepDPM grouped images with similar semantic properties.

Figure 1. Examples from cluster #1

Figure 2. Examples from cluster #2

Figure 3. Examples from cluster #3

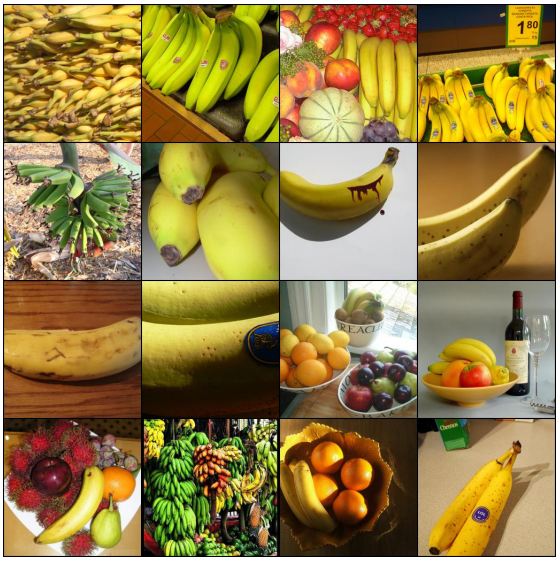


Figure 4. Examples from cluster #4

Figure 5. Examples from cluster #5

Figure 7. Examples from cluster #7

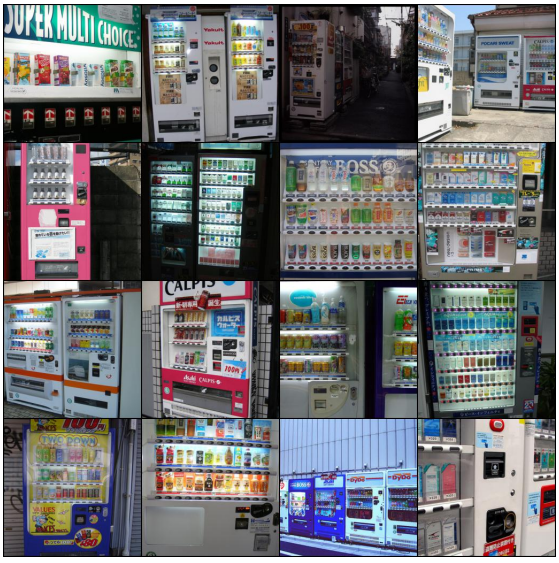


Figure 6. Examples from cluster #6

Figure 8. Examples from cluster #8

Method	Inferred K		
	MNIST^{imb}	USPS^{imb}	Fashion-MNIST^{imb}
DBSCAN	9.0±0.00	6.0±0.00	4.0±0.00
DPM Sampler	11.9±0.32	7.3±0.48	11.6±0.97
moVB	13.6±0.80	11.2±1.33	17.8±2.27
DeepDPM (Ours)	**10.3±0.44**	**9.1±0.22**	**9.4±0.52**

Table 1. Comparing the mean inferred value (±std. dev.) K value, across 10 runs, among the competing nonparametric methods. GT $K = 10$. The symbol ^{imb} marks imbalanced datasets.

1.2. Comparison with Classical Clustering Methods: the Inferred K Value in the Imbalanced Case

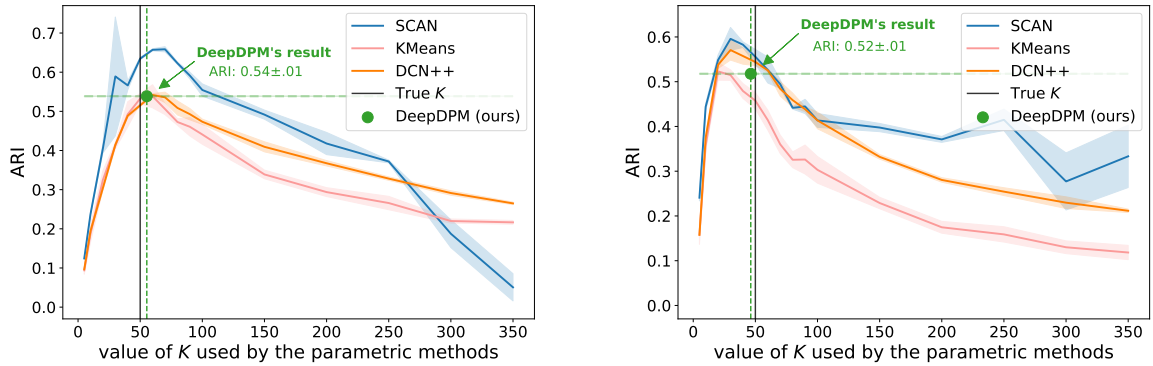
In the paper, we showed how DeepDPM outperforms classical (*i.e.* non-deep) clustering methods, including K-means, GMM, DBSCAN [7], moVB [9] and a State-Of-The-Art (SOTA) DPM sampler [6]. We also showed how, in the balanced case, DeepDPM inferred a more accurate estimate of K. Here, to complete the picture, we provide the results for inferring K in the imbalanced setting. In this case too, DeepDPM inferred the most accurate K value; see Table 1.

1.3. When Parametric Methods Break: the ARI and NMI Metrics

In the paper, we investigated how parametric methods operate with an unknown K, on both balanced and imbalanced datasets. The parametric methods we compared with were K-means, DCN++, an improved variant of DCN [16], and SCAN [14]. As the reader may recall, we ran the parametric methods (which require specifying K) with different K values ranging between 5 and 350 (the exact values we used were: 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 250, 300, and 350). As DeepDPM is a nonparametric method, instead of using a known fixed value of K, it infers it. Specifically, on the ImageNet-50 dataset, DeepDPM inferred $K = 55.3 \pm 1.53$ (the mean and std. dev. across 3 different runs) in the balanced case and 46.3 ± 2.52 in the imbalanced one. In both cases, our results were fairly close to the GT value ($K = 50$).

Here we provide the complementary metrics which were not shown in the paper due to page limits: the ARI (see Figure 9) and NMI (see Figure 10) metrics.

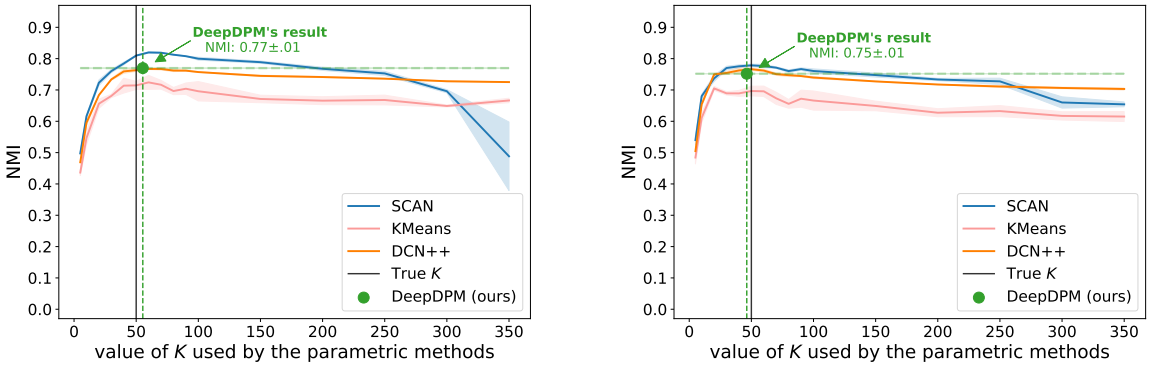
While the ARI (similarly to clustering accuracy) penalizes over- and under-clustering similarly, the NMI metric is not as sensitive to over-clustering as it is to under-clustering. For more details, see § 2.1.2. Thus, the NMI score of parametric methods remains relatively-stable when $K \in [50, 250]$ for both the balanced case and the imbalanced one. Moreover, as can be seen in Figure 10a, for the parametric SOTA method, SCAN, the NMI misleadingly peaks at $K = 70$ (recall the true K is 50). Despite not having access to the additional information used (and required) by the parametric methods – that is, the value of K – and despite the fact that NMI is relatively insensitive to over-clustering, DeepDPM still reaches comparable performance to the parametric methods in both the ARI and NMI metrics, especially in the imbalanced case.



(a) ImageNet50: The original balanced dataset

(b) ImageNet50: An imbalanced dataset

Figure 9. Mean ARI of 3 runs (± std. dev.) on 50 classes of ImageNet.



(a) ImageNet50: The original balanced dataset

(b) ImageNet50: An imbalanced dataset

Figure 10. Mean NMI of 3 runs (± std. dev.) on 50 classes of ImageNet. Note that the NMI metric is not sufficiently sensitive to over clustering; *e.g.*, in the balanced case, SCAN's NMI peaks around $K = 70$ while the true K is 50.

1.4. Ablation study: NMI, ARI and the Final Value of K

We provide in Table 2 the ARI, NMI and final K values for the ablation study described in the paper. The ACC values already appear in the paper.

	NMI			ARI			final K		
	K_{init}=3	K_{init}=10	K_{init}=30	K_{init}=3	K_{init}=10	K_{init}=30	K_{init}=3	K_{init}=10	K_{init}=30
No splits/merges	.53±.00	.67±.01	.64±.00	.22±.00	.49±.02	.43±.01	3	10	30
No splits	.53±.00	.67±.01	.63±.00	.22±.00	.49±.02	.42±.02	3±.00	**10±.00**	23±.00
No merges	.61±.00	.66±.00	.64±.01	.38±.00	.48±.01	.44±.01	5±.00	**10±.00**	21.3±1.53
2-means instead of f_{sub}	**.68±.00**	**.68±.01**	.67±.00	**.50±.00**	**.51±.02**	.48±.01	11±.00	12±.00	14.67±.58
No priors in the M step	.65±.00	.66±.01	.66±.00	.48±.01	.48±.02	.50±.00	12±1.73	14±1.41	**13.67±.58**
Isotropic loss instead of $\mathcal{L}_{\text{cl}}$	.67±.01	.67±.00	.67±.00	**.50±.01**	.49±.00	.49±.00	**10±0.82**	9±.00	9.25±.50
DeepDPM (full method)	**.68±.00**	.67±.01	**.68±.01**	**.50±.00**	**.51±.01**	**.52±.01**	10.67±.58	11.67±1.15	14±.00

Table 2. DeepDPM's performance under different ablations.

2. Metrics and Datasets Used in the Evaluation

2.1. Evaluation Metrics

In our evaluations we used three common supervised clustering metrics: clustering accuracy (ACC), Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI). The ACC and NMI range between 0 and 1, and the ARI ranges between -1 and 1. For all metrics the higher the better and all of them can accommodate for different numbers of classes between the prediction and the ground truth. We also used the silhouette score, which is an unsupervised metric, in order to find (in an unsupervised way) the best K for the parametric methods. The silhouette score ranges between -1 and 1 (the higher the better).

2.1.1 ACC

ACC is defined by:

$$\text{ACC} = \max_{m} \left(\frac{\sum_{i=1}^{N} \mathbb{1}(y_i = m(z_i))}{N} \right) \tag{1}$$

where N is the number of data points, y_i is the Ground-Truth (GT) class label of data point i, z_i is the predicted cluster assignment according to the clustering algorithm under consideration, $\mathbb{1}(\cdot)$ is the indicator function, and m is defined by all possible one-to-one mappings between the predicted class membership and the ground-truth one.

Thus, this metric can be compared to the standard accuracy measure used in the supervised-learning settings, with class mapping, where the mapping used is the best match between the GT classes and the predicted ones. To find the best match, we use the popular Hungarian matching algorithm.

2.1.2 NMI

Let $\boldsymbol{z} = (z_i)_{i=1}^N$ and let $\boldsymbol{y} = (y_i)_{i=1}^N$. NMI is defined by:

$$\text{NMI} = \frac{2 \times I(\boldsymbol{y}; \boldsymbol{z})}{H(\boldsymbol{y}) + H(\boldsymbol{z})} \tag{2}$$

where $H(.)$ stands for entropy and $I(.;.)$ denotes Mutual Information (MI). One problem with this metric, however, is that the MI term, which appears in the numerator, does not penalize large cardinalities (*i.e.*, over clustering). The denominator partially fixes this, but not entirely. Thus, NMI is not sensitive enough to over clustering. See for example Figure 10.

2.1.3 ARI

The Rand index (RI) quantifies the percentage of "correct" decisions for each pair of data points. A decision is correct if two examples either belong to the same GT class and the same cluster assignment (a true positive, TP), or being from two different GT classes and assigned to two different clusters (a true negative, TN). Similarly, clustering errors are false positives (FP) and false negatives (FN). Then RI is computed by:

$$RI = \frac{TP + TN}{TP + TN + FP + FN}. \tag{3}$$

The ARI measure is the corrected-for-chance version of the Rand index. Given a set $\mathcal{S}$ of N elements, and two groupings or partitions (e.g. $\boldsymbol{y}$ and $\boldsymbol{z}$) of these elements, the overlap between $\boldsymbol{y}$ and $\boldsymbol{z}$ can be summarized in a contingency table $[c_{kl}]$ where each entry c_{kl} denotes the number of objects in common between y_k and z_k : $c_{kl} = |y_k \cap z_k|$. Let a_k be the sum of each row, meaning, $a_k = \sum_l c_{kl}$, and b_k the sum of each column, *i.e.* $b_k = \sum_k c_{kl}$.

The ARI measure is then calculated by:

$$ARI = \frac{\sum_{kl} \binom{n_{kl}}{2} - [\sum_k \binom{a_k}{2} \sum_l \binom{b_l}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_k \binom{a_k}{2} + \sum_l \binom{b_l}{2}] - [\sum_k \binom{a_k}{2} \sum_l \binom{b_l}{2}]/\binom{n}{2}} \tag{4}$$

2.1.4 Silhouette Score

So far we have discussed supervised evaluation scores, meaning, ones that require the GT labels. However, in unsupervised cases where the GT is unknown in training, one often needs a metric to evaluate the model; *e.g.* when performing hyperparameter tuning or model selection for parametric methods. In this case, usually a parametric model is run using a range of different values for K, and an unsupervised criterion is used to choose the best model (best value for K).

One of the most common unsupervised clustering metrics is the silhouette score, which quantifies the clustering quality by measuring the amount of "cohesion" within a cluster, and "separation" between different clusters. Meaning, the more data points within each cluster are closely-packed and different clusters are well-separated, the higher the silhouette score is. More formally, given data $\mathcal{X} = (\boldsymbol{x}_i)_{i=1}^N \in \mathcal{R}^{N \times d}$ and its clustering prediction $\boldsymbol{z}$, for data point $\boldsymbol{x}_i$ with cluster label k ($z_i = k$), let

$$a(i) = \frac{1}{|N_k| - 1} \sum_{\boldsymbol{x}_j : z_j = k} d(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{5}$$

be the mean distance between $\boldsymbol{x}_i$ and all other data points in the same cluster, where $|N_k|$ is the number of points hard assigned to cluster k, and and $d(i, j)$ is the distance between data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

Let

$$b(i) = \min_{k':k' \neq k} \frac{1}{N_{k'}} \sum_{\boldsymbol{x}_j : z_j = k'} d(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{6}$$

be the smallest mean distance of datapoint $\boldsymbol{x}_i$ to all points in any other cluster, of which $\boldsymbol{x}_i$ is not a member.
Now, the silhouette score of $\boldsymbol{x}_i$ is defined as:

$$s(i) = \begin{cases} \frac{b(i)-a(i)}{\max(a(i),b(i))}, & \text{if } N_k > 1 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Thus, $s(i) \in [-1, 1]$. Finally, the total silhouette score is the average of all values for $s(i)$. Note that the silhouette score does not use the GT labels, and thus it is an unsupervised metric.

2.2. Datasets for Evaluation

Datasets. We evaluate our method on text and image datasets in varying scales. The summary statistics on the datasets are available in Table 3.

Dataset	Train samples	Val samples	Data Dimension	GT K
MNIST [5]	60,000	10,000	28×28	10
USPS [10]	7291	2007	16×16	10
Fashion- MNIST [15]	60,000	10,000	28×28	10
STL10 [3]	5,000	8,000	$96 \times 96 \times 3$	10
Reuters10K [11]	10000	-	28×28	4
ImageNet-50	64,274	2,500	$224 \times 224 \times 3$	50
ImageNet [4]	1,281,167	50,000	$224 \times 224 \times 3$	1000

Table 3. Descriptive properties of the datasets used for evaluation.

Remarks regarding the ImageNet and ImageNet-50 datasets. The creators of ImageNet [4] do not hold the copyright of all images, and the usage of that dataset is governed by the terms of the ImageNet license `https://image-net.org/download`. ImageNet-50 is a subset of 50 randomly-selected classes of ImageNet, curated by [14].

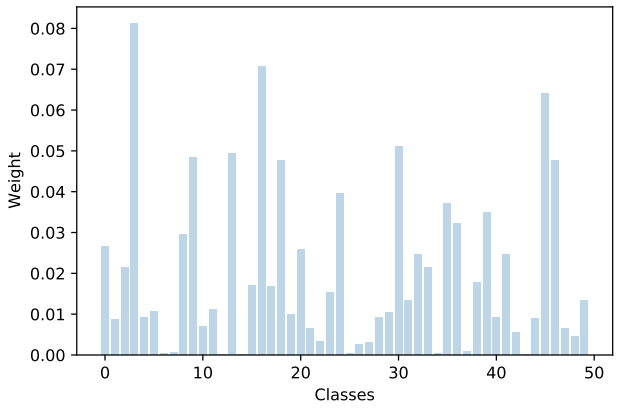
2.3. Train and Validation Sets

In general, we train and evaluate using the train and validation split respectively for most of the comparisons. However, when comparing with deep nonparametric methods (which we had problems to run their code and had to resort to use their reported results), to allow for a fair comparison, we computed the evaluation metrics on the entire dataset (as this is what their reported numbers referred to), meaning, combining the train and validation sets into one set. Note that in this case too, we still trained our model only on the training set. Also recall the training (of both our method and the competitors) is unsupervised and used neither the GT labels nor the GT K.

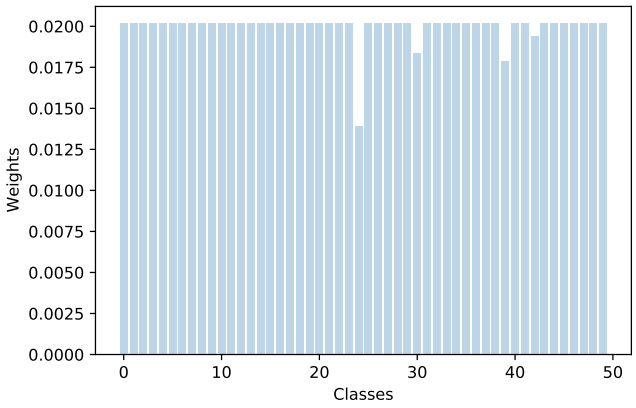
2.4. Creating Imbalanced Datasets

To create imbalanced datasets for the smaller datasets, *i.e.*, MNIST, USPS and Fashion-MNIST, we undersampled some of the classes using random proportions. Concretely, for MNIST and USPS we sampled 10%, 5%, 20%, and 30% of the total amount of examples of classes 8, 2, 5, and 9, respectively. All the other classes were used in full. For Fashion-MNIST dataset, classes 0, 3, 5, 7, and 8 were under-sampled with 37%, 19%, 41%, 54%, 19% of the total number of examples per class (the classes and percentages were chosen randomly).

To create an imbalanced version of ImageNet-50, we sampled a 50-bin normalized histogram from a uniform Dirichlet distribution (not to be confused with a Dirichlet process) over the 50-dimensional probability simplex. This means that any histogram was equally probable (not to be confused with a uniform histogram). The random nonuniform histogram we sampled is shown in Figure 11a. For comparison, Figure 11b shows the original class distribution of ImageNet-50 which is almost perfectly balanced (*i.e.*, almost uniform).



(a) The random histogram we sampled from the uniform distribution over the 50-dimensional probability simplex. This histogram was used for creating the imbalanced version of ImageNet-50.



(b) The class histogram of the original ImageNet-50 dataset.

Figure 11. The balanced vs. imbalanced histograms for ImageNet-50.

3. The NIW Prior, Marginal Data Likelihood, the Weighted Bayesian Estimates, and the Concentration Parameter

To make our work self-contained, below we provide the details for the key Bayesian calculations that we use. For more details about the known theoretical results in § 3.1, § 3.2, and § 3.4, see [1].

3.1. The Normal Inverse Wishart Distribution

In the Dirichlet Process Gaussian Mixture Model (DPGMM), like in the Bayesian GMM, each component's parameters, $(\boldsymbol{\theta}_k, \pi_k)$, where $\boldsymbol{\theta}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ denote the mean and covariance matrix, and π_k is the mixture weight, are assumed to be drawn from a certain prior distribution. A common choice for a prior for $\boldsymbol{\theta}_k$ is the Normal-Inverse-Wishart (NIW) distribution. This is because the latter is a conjugate priorto the multivariate normal distribution with an unknown mean and an unknown covariance matrix. The conjugacy property guarantees that the posterior probability will be in the same distribution family as the NIW prior, and provides a closed-form expression for it, which is algebraically convenient for inference.

The probability density function (pdf) of the Inverse-Wishart (IW) distribution over $d \times d$ Symmetric and Positive Definite (SPD) matrices, evaluated at the $d \times d$ SPD matrix $\boldsymbol{\Sigma}_k$, is

$$\mathcal{W}^{-1}(\boldsymbol{\Sigma}_k; \nu, \boldsymbol{\Psi}) = \frac{|\nu\boldsymbol{\Psi}|^{\frac{\nu}{2}}}{2^{\frac{\nu d}{2}} \Gamma_d(\frac{\nu}{2})} |\boldsymbol{\Sigma}_k|^{-\frac{\nu+d+1}{2}} e^{-\frac{1}{2}\operatorname{tr}(\nu\boldsymbol{\Psi}\boldsymbol{\Sigma}_k^{-1})} \tag{8}$$

where $\nu > d-1$, $\boldsymbol{\Psi} \in \mathbb{R}^{d\times d}$ is SPD, and Γ_d is the (d-dimensional) multivariate gamma function. Equivalently, we may write

$$\boldsymbol{\Sigma}_k \sim \mathcal{W}^{-1}(\nu, \boldsymbol{\Psi})\,. \tag{9}$$

The positive real number ν and the SPD matrix $\boldsymbol{\Psi}$ are called the hyperparameters of the IW distribution.

Now let $\boldsymbol{\mu}_k \in \mathbb{R}^d$. The vector $\boldsymbol{\mu}_k$ and the SPD matrix $\boldsymbol{\Sigma}_k$ are said to be Normal-Inverse-Wishart distributed if their joint pdf is

$$p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k; \kappa, \boldsymbol{m}, \nu, \boldsymbol{\Psi}) = \mathrm{NIW}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k; \kappa, \boldsymbol{m}, \nu, \boldsymbol{\Psi}) \triangleq \overbrace{\mathcal{N}(\boldsymbol{\mu}_k; \boldsymbol{m}, \tfrac{1}{\kappa}\boldsymbol{\Sigma}_k)}^{p(\boldsymbol{\mu}_k|\boldsymbol{\Sigma}_k;\kappa,\boldsymbol{m})} \overbrace{\mathcal{W}^{-1}(\boldsymbol{\Sigma}_k; \nu, \boldsymbol{\Psi})}^{p(\boldsymbol{\Sigma}_k;\nu,\boldsymbol{\Psi})} \tag{10}$$

where $\boldsymbol{m} \in \mathbb{R}^d$ and $\kappa > 0$ (while ν and $\boldsymbol{\Psi}$ are as before) and $\mathcal{N}(\boldsymbol{\mu}_k; \boldsymbol{m}, \frac{1}{\kappa}\boldsymbol{\Sigma}_k)$ is a d-dimensional Gaussian pdf, evaluated at $\boldsymbol{\mu}_k$, with mean $\boldsymbol{m}$ and covariance $\frac{1}{\kappa}\boldsymbol{\Sigma}_k$.

Equivalently, we may write

$$(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \sim \mathrm{NIW}(\boldsymbol{m}, \kappa, \boldsymbol{\Psi}, \nu)\,. \tag{11}$$

The elements of the tuple $\lambda \triangleq (\boldsymbol{m}, \kappa, \boldsymbol{\Psi}, \nu)$ are called the hyperparameters of the NIW distribution. Particularly, ν and κ are called the pseudocounts of that distribution. Loosely speaking, the higher ν and κ are, the more the distribution is peaked (roughly) around $\boldsymbol{\Psi}$ and around $\boldsymbol{m}$, respectively.

Remark: do not confuse k (the index of the Gaussian/cluster) with κ ("kappa", a hyperparameter of the NIW distribution).

Assuming, for a moment, a hard-assignment setting, let $N_k = |\{i : z_i = k\}|$ and let $\mathcal{X}_k = (\boldsymbol{x}_i)_{i:z_i=k}$ denote N_k i.i.d. draws from $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. The key reason why the NIW distribution is widely used [8] as a prior over the parameters, $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, is conjugacy (to the Gaussian likelihood). Namely, the posterior distribution over these parameters is also NIW,

$$p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k | \mathcal{X}_k) = \mathrm{NIW}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k; \kappa^*, \boldsymbol{m}_k^*, \nu^*, \boldsymbol{\Psi}_k^*)\,, \tag{12}$$

and its so-called posterior hyperparameters are given in closed form:

$$\kappa_k^* = \kappa + N_k \tag{13}$$

$$\boldsymbol{m}_k^* = \frac{1}{\kappa_k^*}\left[\kappa \boldsymbol{m} + \sum_{i:z_i=k} \boldsymbol{x}_i\right] \tag{14}$$

$$\nu_k^* = \nu + N_k \tag{15}$$

$$\boldsymbol{\Psi}_k^* = \frac{1}{\nu^*}\left[\nu\boldsymbol{\Psi} + \kappa \boldsymbol{m}\boldsymbol{m}^T + \left(\sum_{i:z_i=k} \boldsymbol{x}_i \boldsymbol{x}_i^T\right) - \kappa_k^* \boldsymbol{m}_k^* (\boldsymbol{m}_k^*)^T\right]\,. \tag{16}$$

Importantly, when ν and κ are much smaller than N, then the specific choice of $\boldsymbol{\mu}_k$ and $\boldsymbol{\Psi}$ becomes negligible, implying a very weak prior.

3.2. The Marginal Likelihood Function

When marginalizing over the parameters of a Gaussian (*i.e.*, its mean and covariance), one obtains the marginal data likelihood (given the hyperparameters of the NIW prior):

$$f_{\boldsymbol{x}}((\boldsymbol{x}_i)_{i=1}^N; \lambda) = f_{\boldsymbol{x}}((\boldsymbol{x}_i)_{i=1}^N; \boldsymbol{m}, \kappa, \boldsymbol{\Psi}, \nu) = \int p((\boldsymbol{x}_i)_{i=1}^N | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k; \lambda) d(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$
$$= \frac{1}{\pi^{\frac{Nd}{2}}} \frac{\Gamma_d(\nu^*/2)}{\Gamma_d(\nu/2)} \frac{|\nu\boldsymbol{\Psi}|^{\nu/2}}{|\nu^*\boldsymbol{\Psi}_k^*|^{\nu^*/2}} \left(\frac{\kappa}{\kappa^*}\right)^{d/2} \tag{17}$$

where Γ_d is the d-dimensional Gamma function.

3.3. Weighted MAP Estimates of the Parameters

We now provide the details of the M step. More concretely, below we explain how we compute the weighted Maximum-a-Posteriori (MAP) estimates of the clusters' and subclusters' parameters, where the weighting is done according to the output of our deep nets.

Let $\lambda = (\boldsymbol{m}, \kappa, \boldsymbol{\Psi}, \nu)$ be the NIW hyperparams. In the unweighted case, the MAP estimates of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are:

$$\boldsymbol{\Sigma}_k = \frac{\nu^* \boldsymbol{\Psi}_k^*}{\nu^* - d + 1} \tag{18}$$

$$\boldsymbol{\mu}_k = \boldsymbol{m}_k^*\,. \tag{19}$$

In our case, the MAP estimates are obtained in a similar way, but with the following differences. Rather than using hard assignments (as in § 3.1), we use soft assignments; *i.e.*, the MAP estimates take all N points into consideration, but with an appropriate weighting of each point. This is nearly identical to the weighted MAP estimates in the standard computation of the

M step in Bayesian EM-GMM, except that here the weighting is done using the $(r_{i,k})$ values (namely, the soft assignments which are our deep net's output). That is, we still use Eq. (18) and Eq. (19), but instead of the posterior hyperparameters from Eqs. (13)–(16), we use their weighted versions (where the weights are the $(r_{i,k})$ values).

$$\kappa_k^* = \kappa + \sum_{i=1}^{N} r_{i,k} \tag{20}$$

$$\boldsymbol{m}_k^* = \frac{1}{\kappa_k^*}\left[\kappa\boldsymbol{m} + \sum_{i=1}^{N} r_{i,k}\boldsymbol{x}_i\right] \tag{21}$$

$$\nu_k^* = \nu + \sum_{i=1}^{N} r_{i,k} \tag{22}$$

$$\boldsymbol{\Psi}_k^* = \frac{1}{\nu^*}\left[\nu\boldsymbol{\Psi} + \kappa\boldsymbol{m}\boldsymbol{m}^T + \left(\sum_{i=1}^{N} r_{i,k}\boldsymbol{x}_i\boldsymbol{x}_i^T\right) - \kappa_k^*\boldsymbol{m}_k^*(\boldsymbol{m}_k^*)^T\right] . \tag{23}$$

The parameters of the subclusters are updated in a very similar way, except that the soft subcluster assignments (*i.e.*, $(\widetilde{r}_{i,j})$) are used instead of the soft cluster assignments.

3.4. The Concentration Parameter of the Dirichlet Process

The concentration parameter of the Dirichlet Process, $\alpha > 0$, is a user-defined hyperparameter that, when sampling from the *prior*, controls (the expected number of) the number of clusters. In short, the higher α is, the more clusters are expected. However, when doing *posterior calculations*, if $\alpha \ll N$, where N is the number of data points, then the importance of α diminishes. Particularly, when computing the Hastings ratios for the splits and merges, the importance of $\alpha \ll N$ is usually negligible.

4. The Factors Affecting K; Our Weak Prior

In a DPGMM, the number of clusters is affected not only by α and the data but also the NIW prior. For example, if ν is very high and $\boldsymbol{\Psi}$ is small, then the model will favor small clusters and thus K is likely to be high (so the small clusters will efficiently cover the data). Conversely, if ν is very hight and $\boldsymbol{\Psi}$ is large, then the model will favor large clusters so K will tend to be small (only few large clusters can cover the entire data). However, we emphasize that in all cases, our choices (see below) for the values of the NIW hyperparameters and the concentration parameter α correspond to a very weak prior. That is, our α, ν and κ are all much smaller than N in all the datasets we experimented with: across all the datasets, the smallest N was 13,000 (in STL-10). As a result, the main factor in determining the inferred K is the data itself.

5. Merge Proposals

In the paper, we explained how K is changed via splits and merges, and described how to compute the acceptance probability of split proposals. Here, we provide the complementary details on merge proposals. In a merge step, we sequentially propose pairs of clusters to merge. To avoid sequentially considering all possible merges, we consider (sequentially) the merges of each cluster with only its 3 nearest neighbors.

The proposal to merge a pair of clusters, k_1 and k_2, is a accepted with probability $(1, H_{\mathrm{m}})$, where

$$H_{\mathrm{m}} = \frac{1}{H_{\mathrm{s}}} = \frac{\Gamma(N_{k_1} + N_{k_2}) f_{\boldsymbol{x}}(\mathcal{X}_{\{k_1,k_2\}}; \lambda)}{\alpha\Gamma(N_{k_1}) f_{\boldsymbol{x}}(\mathcal{X}_{k_1}; \lambda)\Gamma(N_{k_2}) f_{\boldsymbol{x}}(\mathcal{X}_{k_2}; \lambda)} \tag{24}$$

is the Hastings ratio, Γ is the Gamma function, $N_{k,1}$ and $N_{k,2}$ are the number of points in clusters 1 or 2, respectively, $\mathcal{X}_k = (\boldsymbol{x}_i)_{i:z_i=k}$ denotes the points in cluster k, and $\mathcal{X}_{\{k_1,k_2\}} = (\boldsymbol{x}_i)_{i:z_i\in\{k_1,k_2\}}$ denotes the points in clusters k_1 and k_2. As for $f_{\boldsymbol{x}}(\cdot;\lambda)$, this is the *marginal* likelihood where λ represents the NIW hyperparameters.

6. Feature Extraction

Here we provide more details on the feature-extraction process. In general, there are two main paradigms: an end-to-end approach in which the features and clustering are learned simultaneously, and a two-step approach in which clustering is performed on pre-computed latent features.

6.1. End-to-end Approach: Jointly Learning Features and Clustering

Here, we loosely follow DCN [16] and similarly start by training an Autoencoder (AE) with a reconstruction loss,

$$\mathcal{L}_{\mathrm{AE}_{\mathrm{recon}}} = \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}_i)) - \boldsymbol{x}_i\|_{\ell_2}^2 \tag{25}$$

where $\boldsymbol{f}$ is the encoder and $\boldsymbol{g}$ is the decoder. Then, while DCN performs K-means on the resulted embeddings to obtain initial cluster centers and assignments, we use our more flexible DeepDPM (which, unlike K-means, assumes neither isotropic classes, uniform weights, nor a known K). Next, we utilize the pipeline of [16] and refine the AE's latent space by training it with both $\mathcal{L}_{\mathrm{recon}}$ and an additional clustering loss:

$$\mathcal{L}_{\mathrm{AE}_{\mathrm{clus}}} = \|\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{\mu}_{z_i}\|_{\ell_2}^2, \tag{26}$$

where z_i is the cluster assignment of $\boldsymbol{x}_i$, and $\boldsymbol{\mu}_{z_i}$ is the cluster's center. This loss encourages $\boldsymbol{f}$ to create small intra-class distances in the latent space. The overall loss is $\mathcal{L}_{\mathrm{AE}_{\mathrm{recon}}} + \frac{\beta}{2}\mathcal{L}_{\mathrm{AE}_{\mathrm{clus}}}$ where $\beta > 0$ is a tunable parameter. While in [16] new cluster assignments and centers are computed after each epoch of the AE, we keep them fixed during this stage, changing only the embeddings. While the pipeline suggested in [16] ends here, we add an alternation scheme where we alternate between clustering the updated embeddings using DeepBNP (keeping the AE frozen) and training the AE (*i.e.*, perform feature learning, while keeping the clustering fixed). We repeat this process several times. Intuitively, during training, DeepDPM is likely to change K, thus, adapting the embeddings accordingly may reveal inter- and intra-class structures which can be useful, in turn, for the clustering module to find meaningful clusters.

6.2. Two-Step Approach: Training on Latent Features.

Another approach for deep clustering is using a pretrained feature extractor backbone. As our method is DL-based, it is easy to concatenate any DL backbone before our DNN. Thus, we follow [14] and use MoCo [2] for (unsupervised) feature extraction.

We provide the specific values we used for the feature extractions below in § 7.

7. Implementation Details and Hyperparameters

We detail here all the training configurations and hyperparameters used in our experiments.

For all the experiments, our clustering net used the following MLP architecture where the MLP had an input layer, a single hidden layer, and an output layer. The number of neurons in the input layer was d (the dimension of the input to the clustering module). The number of hidden units was always 50 in all our experiments (changing that number had little effect on the results). The (changing) number of neurons in the output layer was K (which corresponds to the changing number of clusters). In addition, in all our experiments we used a batch size of 128, a learning rate (lr) of 0.0005 for the clustering net, and an lr of 0.005 for the subclustering nets. As for the prior hyperparams, for the DP's α we chose $\alpha = 10$, and for the NIW hyperparams we used $\kappa = 0.0001$, set $\boldsymbol{m}$ to be the data mean, and ν to be $d + 2$. We used different $\boldsymbol{\Psi}$ values in each experiment, as detailed below.

Below in § 7.4 we give some guidelines on how to choose the key hyperparameters for DeepDPM.

7.1. Setup Used for Comparing with Classical Methods

Feature Extraction. For this experiment, we used the feature extraction procedure suggested by [12] where we first trained a deep Autoencoder (AE) and then transformed its latent space using a UMAP transformation [13]. We used the same configurations as in [12].

DeepDPM hyperparameters. For all experiments, we initialized DeepDPM with $K = 1$, DeepDPM was trained for 500 epochs. We set $\boldsymbol{\Psi} = \boldsymbol{I} \times 0.005$ in all the datasets, where $\boldsymbol{I}$ denotes the identity matrix. Note that we used the same configuration for the three datasets, in both the balanced and imbalanced cases.

7.2. Training on Latent Features

Here, we give the hyperparameters we used for evaluating our method on STL-10 and ImageNet. For both datasets we used the unsupervised pretrained feature extractor MoCo [2] and trained DeepDPM on top of the resulting features. For STL-10 we pretrained it for 1000 epochs (on STL-10's train set) and for ImageNet we used the pretrained weights available online.

Dataset	AE architecture	AE lr	DeepDPM training epochs
MNIST	D-500-500-2000-10	0.002	200
Reuters10K	D-500-500-2000-75	0.002	300
ImageNet-50	D-500-500-2000-10	0.002	300

Table 4. Implementation details for DeepDPM's experiments. D denotes the input dimension. For all datasets but ImageNet-50 it is the original data dimension. For ImageNet-50, $D = 128$, the output dimension of MoCo [2]

For STL-10, we initialized DeepDPM with $K = 3$ and trained it for 500 epochs with $\mathbf{\Psi} = \boldsymbol{I} \times 0.05$. For ImageNet, we initialized DeepDPM with $K = 150$ and trained it for 200 epochs with $\mathbf{\Psi} = \boldsymbol{I} \times 0.001$.

7.3. Jointly Learning Features and Clustering

As described in the paper, we adapted the feature learning pipeline from DCN [16] to jointly learn features and clustering in alternation. Table 4 shows the AE architectures, the lr values, and the number of DeepDPM epochs. For ImageNet-50, we trained an AE on top of the features generated by MoCo [2]. For all other datasets, it was trained on top of the original data dimension. For MNIST and Reuters10K we used three alternations, for ImageNet-50 we used two alternations in the balanced case and four alternations for the imbalanced. See below in § 7.4 how we chose the number of alternations.

DeepDPM was initialized with $K = 3$ for MNIST, $K = 1$ for Reuters10K and $K = 10$ for ImageNet-50. As per the prior hyperparams, we chose $\mathbf{\Psi} = 0.005$ for MNIST and Reuters10K. For ImageNet-50 we chose $\mathbf{\Psi} = \boldsymbol{I} \times \mathrm{std}(\mathcal{X}) \times 0.0001$, where $std(\mathcal{X})$ denotes the standard deviation of the data. Note that in both the balanced and imbalanced cases of ImageNet-50 we used the same hyperparameters, where the only difference was the number of alternations.

7.4. General Recommendations for Choosing Hyperparameters

Number of epochs. The number of epochs is chosen by the amount of epochs after which DeepDPM has converged to a certain K; *i.e.*, no more split/merge proposal are accepted. We chose it empirically by training DeepDPM and measuring the average number of epochs it took for K to stabilize. We set the maximal number of epochs to 100 plus that average epoch number.

Number of alternations. When performing feature learning and clustering in alternation, we need to choose the number of times we perform the alternations (one alternation includes training the AE followed by the DeepDPM training). We stop the alternations once the DeepDPM's inferred K stabilizes.

The initial value for K. As we showed, the initial value of K has little effect on the final clustering that DeepDPM generates. Thus, it can be chosen arbitrarily. That said, the value of the initial K can affect the speed of convergence (if DeepDPM starts with a more accurate estimate then less splits and merges will happen and the DeepDPM will stabilize faster). A reasonable choice is to choose the initial K to be proportional to N, the number of datapoints; *e.g.*, $N/10000$. Note that unlike parametric methods, *this is used only as an initialization value which is expected to change*.

Choosing $\mathbf{\Psi}$. As a rule of thumb, we took $\mathbf{\Psi}$ to be proportional to $\boldsymbol{I}$ (*i.e.*, the $d \times d$ identity matrix) with the elements of the main diagonal being the data's standard deviation, scaled by an additional scalar s. The choice of s is based on empirically seeing that a substantial amount of both splits and merges proposals are accepted during the first few hundred epochs. Note that as $\mathbf{\Psi}$ is getting smaller, more splits will be accepted. Thus, if $\mathbf{\Psi}$ is too small, only splits will occur (and no merges) and if it is too large, no splits will be accepted.

References

[1] Jason Chang. *Sampling in computer vision and Bayesian nonparametric mixtures*. PhD thesis, Massachusetts Institute of Technology, 2014. 9

[2] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv:2003.04297*, 2020. 12, 13

[3] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *AISTATS*, 2011. 8

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 8

[5] Li Deng. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 2012. 8

[6] Or Dinari, Angel Yu, Oren Freifeld, and John Fisher III. Distributed MCMC inference in Dirichlet process mixture models using Julia. In *IEEE CCGRID Workshop on High Performance Machine Learning*, 2019. 5

[7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, 1996. 5

[8] Andrew Gelman, Hal S Stern, John B Carlin, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2013. 10

[9] Michael C. Hughes and Erik B Sudderth. Memoized online variational inference for Dirichlet process mixture models. In *NeurIPS*, 2013. 5

[10] Jonathan J Hull. A database for handwritten text recognition research. *IEEE TPAMI*, 1994. 8

[11] David D Lewis, Yiming Yang, Tony Russell-Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *JMLR*, 2004. 8

[12] Ryan McConville, Raul Santos-Rodriguez, Robert J Piechocki, and Ian Craddock. N2d:(not too) deep clustering via clustering the local manifold of an autoencoded embedding. In *ICPR*, 2020. 12

[13] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv:1802.03426*, 2018. 12

[14] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. SCAN: Learning to classify images without labels. In *ECCV*, 2020. 5, 8, 12

[15] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. 08 2017. 8

[16] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *ICML*, 2017. 5, 12, 13