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 **2**
1.1. Additional Visual Examples of Images Clustered Together . . . 2
1.2. Comparison with Classical Clustering Methods: the Inferred K Value in the Imbalanced Case . . . 5
1.3. When Parametric Methods Break: the ARI and NMI Metrics . . . 5
1.4. Ablation study: NMI, ARI and the Final Value of K . . . 6

2. Metrics and Datasets Used in the Evaluation **6**
2.1. Evaluation Metrics . . . 6
2.1.1 ACC . . . 6
2.1.2 NMI . . . 7
2.1.3 ARI . . . 7
2.1.4 Silhouette Score . . . 7
2.2. Datasets for Evaluation . . . 8
2.3. Train and Validation Sets . . . 8
2.4. Creating Imbalanced Datasets . . . 8

3. The NIW Prior, Marginal Data Likelihood, the Weighted Bayesian Estimates, and the Concentration Parameter **9**
3.1. The Normal Inverse Wishart Distribution . . . 9
3.2. The Marginal Likelihood Function . . . 10
3.3. Weighted MAP Estimates of the Parameters . . . 10
3.4. The Concentration Parameter of the Dirichlet Process . . . 11

4. The Factors Affecting K; Our Weak Prior **11**

5. Merge Proposals **11**

6. Feature Extraction **11**
6.1. End-to-end Approach: Jointly Learning Features and Clustering . . . 12
6.2. Two-Step Approach: Training on Latent Features. . . . 12

7. Implementation Details and Hyperparameters **12**
7.1. Setup Used for Comparing with Classical Methods 12
7.2. Training on Latent Features 12
7.3. Jointly Learning Features and Clustering 13
7.4. General Recommendations for Choosing Hyperparameters 13

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

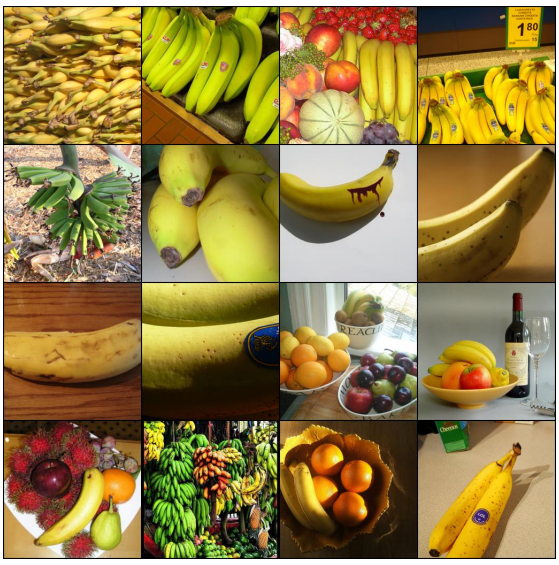


Figure 4. Examples from cluster #4

Figure 5. Examples from cluster #5

Figure 7. Examples from cluster #7

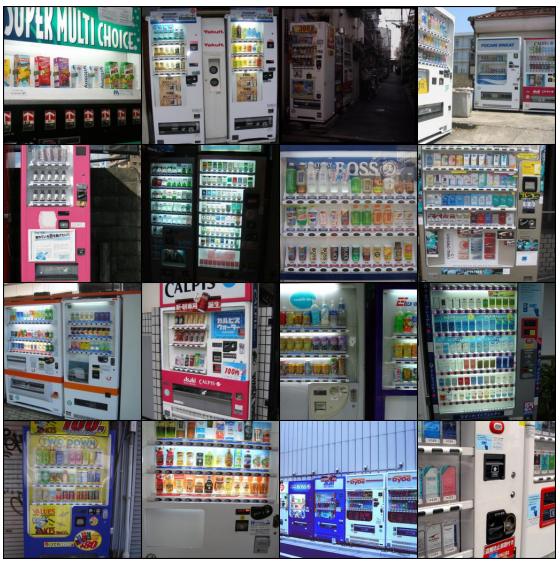


Figure 6. Examples from cluster #6

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

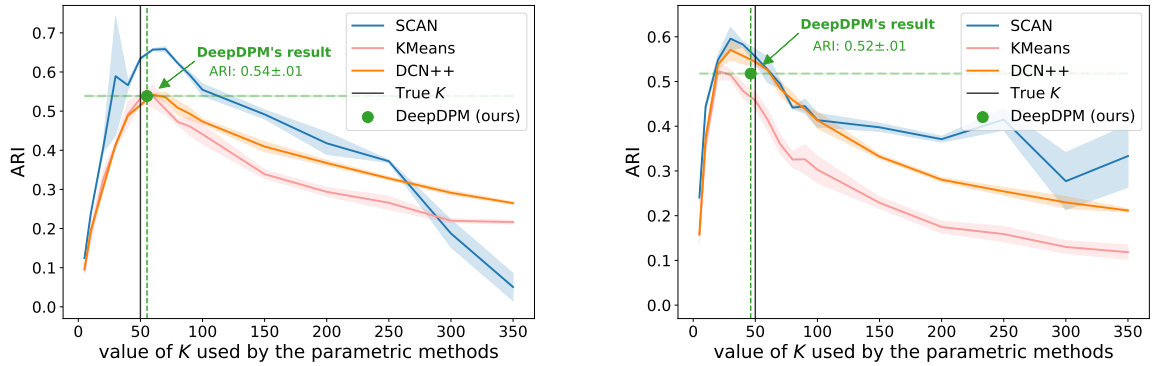
In the paper, we showed how DeepDPM outperforms classical (*i.e.* non-deep) clustering methods, including K-means, GMM, DBSCAN [7], moVB [9] and a State-Of-The-Art (SOTA) DPM sampler [6]. We also showed how, in the balanced case, DeepDPM inferred a more accurate estimate of K. Here, to complete the picture, 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.

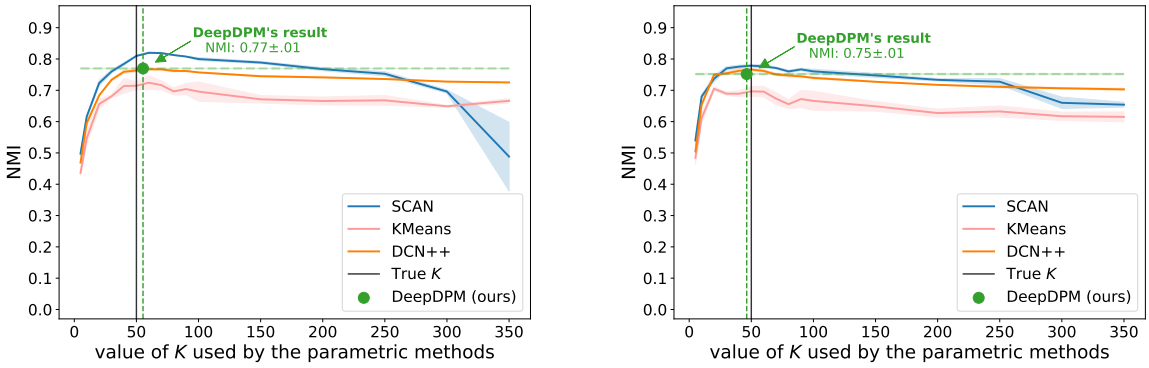
While the ARI (similarly to clustering accuracy) penalizes over- and under-clustering similarly, the NMI metric is not as sensitive to over-clustering as it is to under-clustering. For more details, see § 2.1.2. Thus, the NMI score of parametric methods remains relatively-stable when $K \in [50, 250]$ 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].

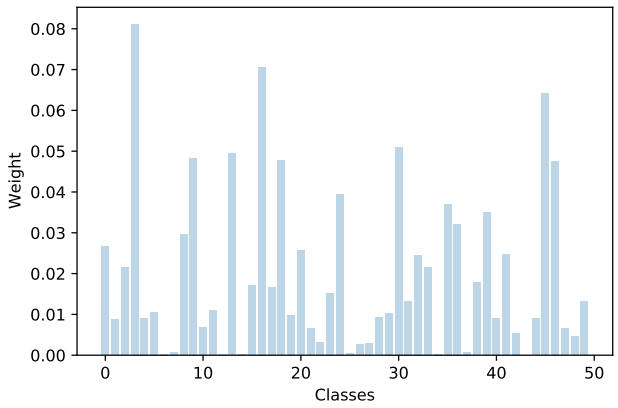
2.3. Train and Validation Sets

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.

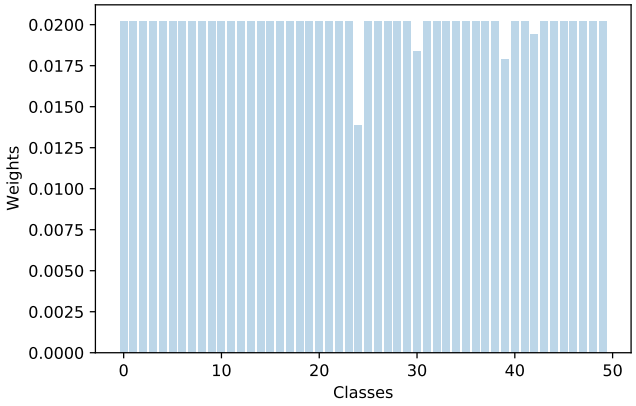
2.4. Creating Imbalanced Datasets

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):

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

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