# Paper Review Seminar

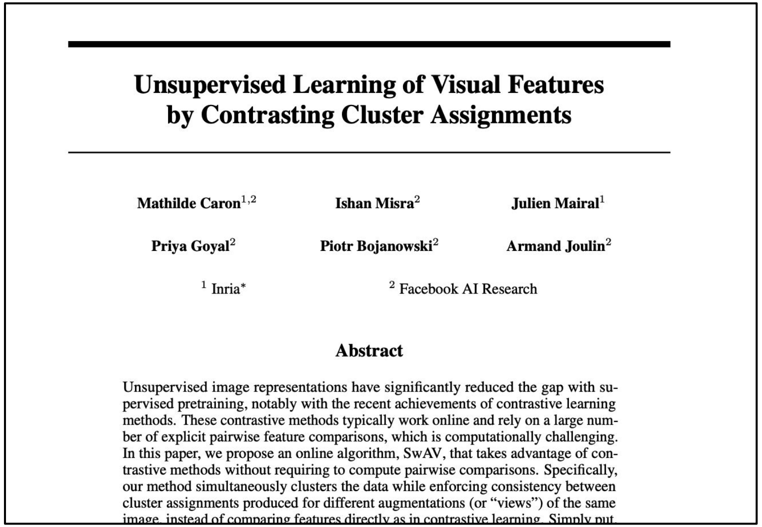
( 2022. 08. 30 )

# [ Clustering Algorithms ]

# SwAV (2021) & DeepDPM (2022)

통계데이터사이언스학과통합과정 4학기 이승한

# Papers

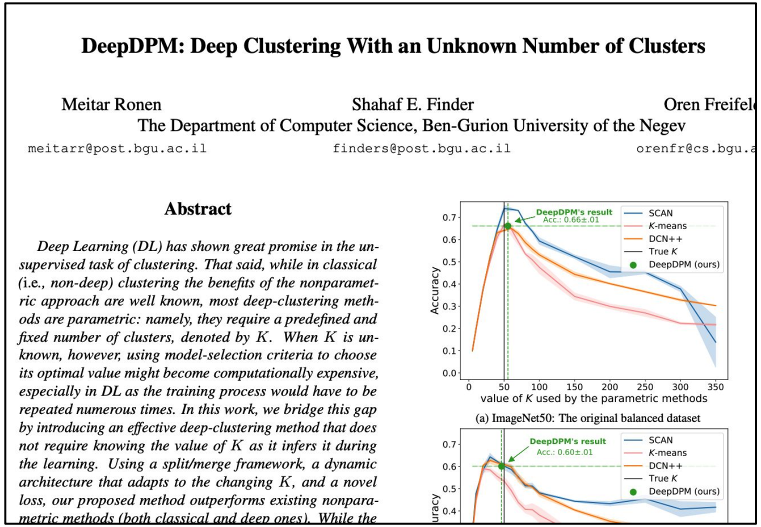

Unsupervised Learning of Visual Features by Contrasting Cluster Assignments

Mathilde Caron[1,2] Ishan Misra[2] Julien Mairal[1]
Priya Goyal[2] Piotr Bojanowski[2] Armand Joulin[2]
[1] Inria* [2] Facebook AI Research

Abstract

Unsupervised image representations have significantly reduced the gap with supervised pretraining, notably with the recent achievements of contrastive learning methods. These contrastive methods typically work online and rely on a large number of explicit pairwise feature comparisons, which is computationally challenging. In this paper, we propose an online algorithm, SwAV, that takes advantage of contrastive methods without requiring to compute pairwise comparisons. Specifically, our method simultaneously clusters the data while enforcing consistency between cluster assignments produced for different augmentations (or "views") of the same image, instead of comparing features directly as in contrastive learning. Simply put,

**SwAV (2021)**


DeepDPM: Deep Clustering With an Unknown Number of Clusters

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

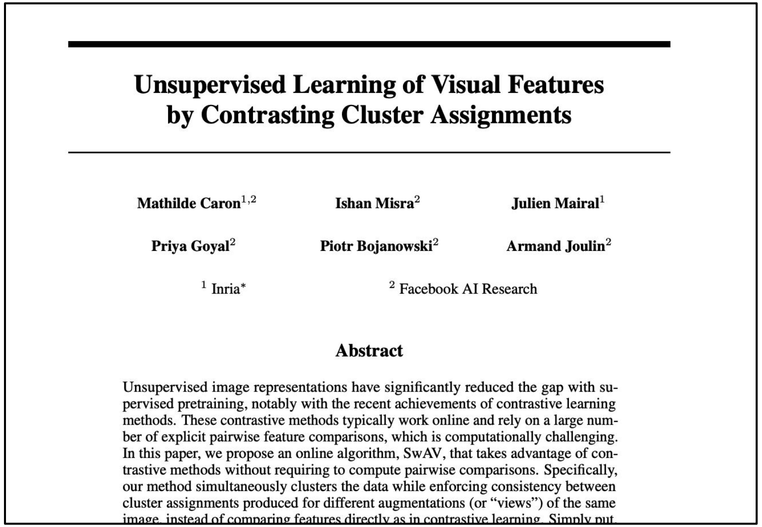
Abstract

*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*


(a) ImageNet50: The original balanced dataset

**DeepDPM (2022)**

# Papers



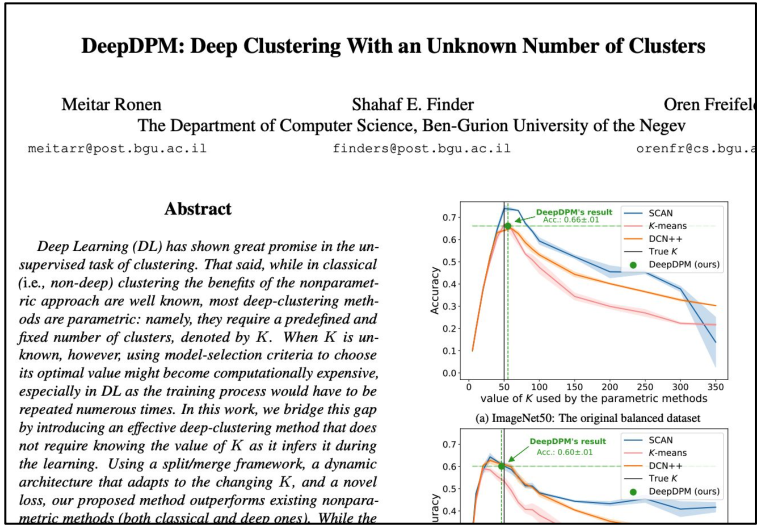
Unsupervised Learning of Visual Features by Contrasting Cluster Assignments

Mathilde Caron[1,2] Ishan Misra[2] Julien Mairal[1]
Priya Goyal[2] Piotr Bojanowski[2] Armand Joulin[2]
[1] Inria* [2] Facebook AI Research

Abstract

Unsupervised image representations have significantly reduced the gap with supervised pretraining, notably with the recent achievements of contrastive learning methods. These contrastive methods typically work online and rely on a large number of explicit pairwise feature comparisons, which is computationally challenging. In this paper, we propose an online algorithm, SwAV, that takes advantage of contrastive methods without requiring to compute pairwise comparisons. Specifically, our method simultaneously clusters the data while enforcing consistency between cluster assignments produced for different augmentations (or "views") of the same image, instead of comparing features directly as in contrastive learning. Simply put,

**SwAV (2021)**



DeepDPM: Deep Clustering With an Unknown Number of Clusters

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

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*



(a) ImageNet50: The original balanced dataset

**DeepDPM (2022)**

# SwAV (2021)

1. Instance Discrimination & Contrastive Loss
2. SwAV
   a. Architecture
   b. Online Clustering
   c. Multi-crop
3. Experiment

# 1. Instance Discrimination & Contrastive Loss

Unsupervised Image Representation, using Contrastive Learning

- **Instance Discrimination task**
  - data augmentations of image A = different **views** of image A
  - each image = each class
- mostly rely on large number of explicit ***PAIRWISE feature comparison***

  → **computationally challenging** !

  * previous works : use random subsets of images / approximate the task ( ex. clustering )

# 1. Instance Discrimination & Contrastive Loss

**Instance Discrimination** rely on combination of 2 elements

- **(1) contrastive loss**
- **(2) set of image transformations**

# 1. Instance Discrimination & Contrastive Loss

**Instance Discrimination** rely on combination of 2 elements

- **(1) contrastive loss**
- **(2) set of image transformations**

→ this paper improves both (1) & (2)

improves (1) by … **online clustering with "swap prediction"**

improves (2) by … **"multi-crop"**

# 2. SwAV

- do not require "**pairwise comparison**"
- simultaneously **clusters the data**, while enforcing **consistency between cluster assignments** produced for different augmentations of same image
- **swapped prediction**
    - predict the "***code of a view***" from the "***representation of another view***"
- **memory efficient**
    - does not require a large memory bank
- propose new data augmentation strategy, **"multi-crop"**

# 2. SwAV

## (1) Architecture

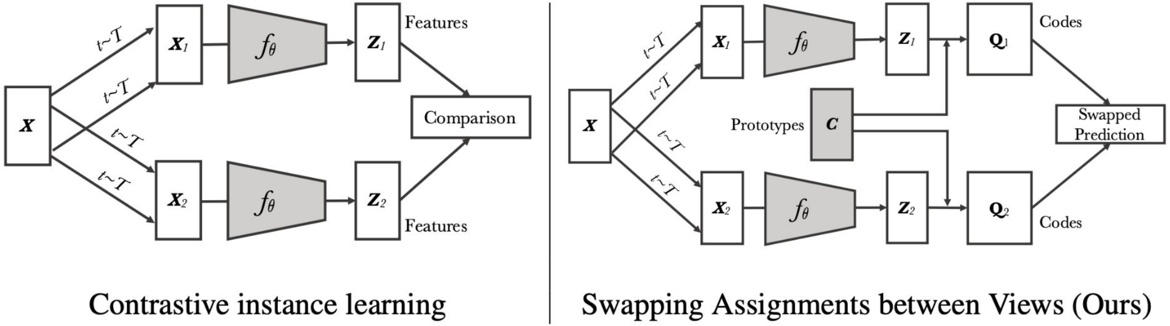



Figure 1: **Contrastive instance learning (left) *vs*. SwAV (right).** In contrastive learning methods applied to instance classification, the features from different transformations of the same images are compared directly to each other. In SwAV, we first obtain "codes" by assigning features to prototype vectors. We then solve a "swapped" prediction problem wherein the codes obtained from one data augmented view are predicted using the other view. Thus, SwAV does not directly compare image features. Prototype vectors are learned along with the ConvNet parameters by backpropragation.

# 2. SwAV

## (1) Architecture

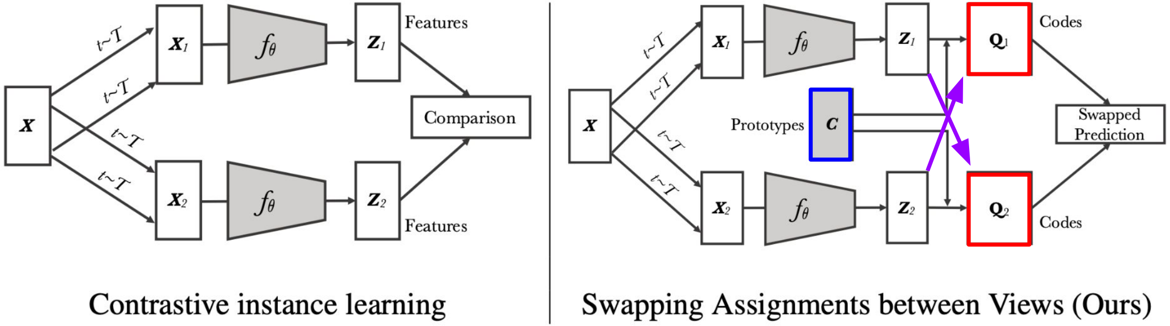



Figure 1: **Contrastive instance learning (left) *vs.* SwAV (right).** In contrastive learning methods applied to instance classification, the features from different transformations of the same images are compared directly to each other. In SwAV, we first obtain "codes" by assigning features to prototype vectors. We then solve a "swapped" prediction problem wherein the codes obtained from one data augmented view are predicted using the other view. Thus, SwAV does not directly compare image features. Prototype vectors are learned along with the ConvNet parameters by backpropragation.

# 2. SwAV

## (1) Architecture

**Compute a code (Q)** from an augmented version of image (Z)

& **predict this code (Q)** from augmented versions of the same image (Z)

Step 1) 2 image features input : $\mathbf{z}_t$ and $\mathbf{z}_s$

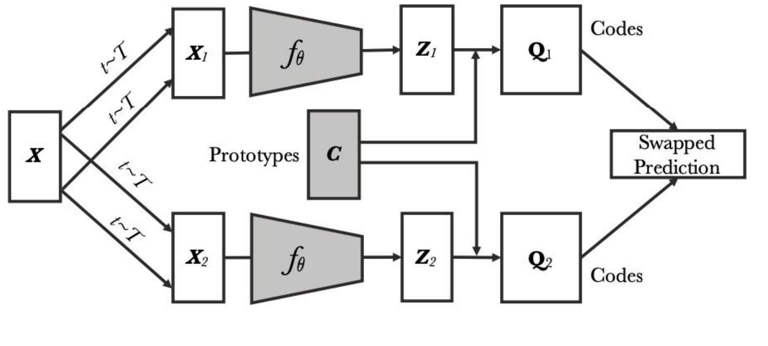
- from different augmentation ( but same image )

Step 2) compute their codes : $\mathbf{q}_t$ and $\mathbf{q}_s$

- by matching these features to a set of $K$ prototypes $\{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$

Step 3) "swapped" prediction problem

- $L(\mathbf{z}_t, \mathbf{z}_s) = \ell(\mathbf{z}_t, \mathbf{q}_s) + \ell(\mathbf{z}_s, \mathbf{q}_t).$
  - $\ell(\mathbf{z}, \mathbf{q})$ : fit between features $\mathbf{z}$ and a code $\mathbf{q}$




Swapping Assignments between Views (Ours)

# 2. SwAV

## (2) Online Clustering

Typical clustering-based methods : **OFFLINE**

→ alternate between (1) cluster assignment & (2) training step

# 2. SwAV

## (2) Online Clustering

Typical clustering-based methods : **OFFLINE**

→ alternate between (1) cluster assignment & (2) training step

**SwAV** ( **Sw**apping **A**ssignments between multiple **V**iews of the same image )

**: learn visual features in an ONLINE fashion ( w.o supervision )**

→ propose an **ONLINE clustering-based SELF-SUPERVISED method**

# 2. SwAV

## (2) Online Clustering

(1) image : $\mathbf{x}_n$

(2) augmented image : $\mathbf{x}_{nt}$ ... applying a transformation $t$

(3) mapped to a vector representation : $\mathbf{z}_{nt} = f_\theta(\mathbf{x}_{nt}) / \| f_\theta(\mathbf{x}_{nt}) \|_2$

(4) compute code : $\mathbf{q}_{nt}$

- by mapping $\mathbf{z}_{nt}$ to a set of $K$ trainable prototype vectors, $\{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$
- $\mathbf{C}$ : matrix whose columns are the $\mathbf{c}_1, \ldots, \mathbf{c}_k$

$\rightarrow$ how to compute these $\mathbf{q}_{nt}$ & update $\{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$ ??    via **Swapped Prediction problem** !!

# 2. SwAV

## (2) Online Clustering

### Loss Function

$$L(\mathbf{z}_t, \mathbf{z}_s) = \ell(\mathbf{z}_t, \mathbf{q}_s) + \ell(\mathbf{z}_s, \mathbf{q}_t).$$

- $\ell(\mathbf{z}_t, \mathbf{q}_s)$ : predicting the code $\mathbf{q}_s$ from the feature $\mathbf{z}_t$
- $\ell(\mathbf{z}_s, \mathbf{q}_t)$ : predicting the code $\mathbf{q}_t$ from the feature $\mathbf{z}_s$

( each term : CE loss )

- $\ell(\mathbf{z}_t, \mathbf{q}_s) = -\sum_k \mathbf{q}_s^{(k)} \log \mathbf{p}_t^{(k)}, \quad \text{where} \quad \mathbf{p}_t^{(k)} = \frac{\exp\left(\frac{1}{\tau}\mathbf{z}_t^\top \mathbf{c}_k\right)}{\sum_{k'} \exp\left(\frac{1}{\tau}\mathbf{z}_t^\top \mathbf{c}_{k'}\right)}$

# 2. SwAV

## (2) Online Clustering

Total Loss ( over all image & pairs of data augmentation )

$$-\frac{1}{N}\sum_{n=1}^{N}\sum_{s,t\sim\mathcal{T}}\left[\frac{1}{\tau}\mathbf{z}_{nt}^{\top}\mathbf{C}\mathbf{q}_{ns}+\frac{1}{\tau}\mathbf{z}_{ns}^{\top}\mathbf{C}\mathbf{q}_{nt}-\log\sum_{k=1}^{K}\exp\left(\frac{\mathbf{z}_{nt}^{\top}\mathbf{c}_k}{\tau}\right)-\log\sum_{k=1}^{K}\exp\left(\frac{\mathbf{z}_{ns}^{\top}\mathbf{c}_k}{\tau}\right)\right]$$

→ optimize w.r.t $\theta$ & $\mathbf{C}$

$\theta$ → **parameter of Feature Extractor**

$\mathbf{C}$ → **Prototype Vectors**

# 2. SwAV

## (2) Online Clustering

Computing Codes **ONLINE** !

$\rightarrow$ compute the codes using only the image features ***within a batch***, using prototypes $\mathbf{C}$

( common prototypes $\mathbf{C}$ are used across different batch )

Induce that all the examples in a batch are ***equally partitioned by the prototypes***

$\rightarrow$ preventing the trivial solution where every image has the same code

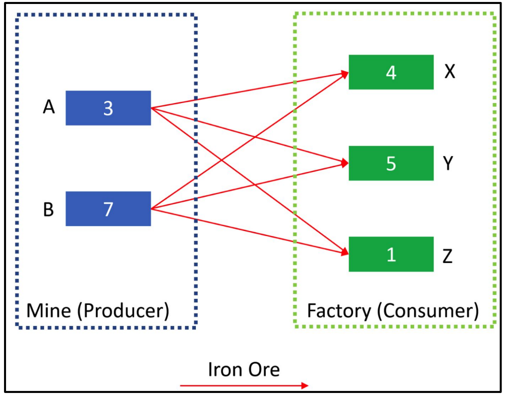
$\longrightarrow$ use **Sinkhorn Algorithm** !!

https://amsword.medium.com/a-simple-introduction-on-sinkhorn-distances-d01a4ef4f085

# 2. SwAV

## (2) Online Clustering

### **Sinkhorn Algorithm**




P(i,j) = Amount of transportation from (i) to (j)

C(i,j) = Cost of ~

$$d = \min \sum_{i,j} P_{i,j} C_{i,j}$$

$$P_{i,j} \geq 0$$

$$\sum_j P_{i,j} = r_i$$

$$\sum_i P_{i,j} = c_j$$

Fig. 2. Optimal transportation problem.

Normalize $r$ & $c$

$\sum r(i) = 1$ & $\sum c(j) = 1$

$\rightarrow$ Interpret $r(i)$ & $c(j)$ as distribution

Total Cost = function of 2 distribution

Use total cost to measure the distance between 2 distribution

https://amsword.medium.com/a-simple-introduction-on-sinkhorn-distances-d01a4ef4f085

# 2. SwAV

## (2) Online Clustering

**Sinkhorn Algorithm**

**Sinkhorn distances**

$$d = \min \sum_{i,j} P_{i,j} C_{i,j}$$

$$P_{i,j} \geq 0$$
$$\sum_j P_{i,j} = r_i$$
$$\sum_i P_{i,j} = c_j$$

$$d^* = \min \sum_{i,j} P_{i,j} C_{i,j}$$

~~$P_{i,j} \geq 0$~~
$$\sum_j P_{i,j} = r_i$$
$$\sum_i P_{i,j} = c_j$$
$$KL(\mathrm{P}|\mathrm{rc}^T) \leq \alpha$$

Fig. 3. Sinkhorn distances d*

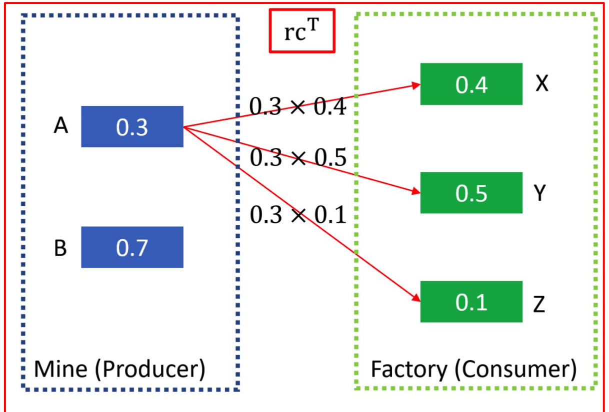



Fig. 4. Transportation plan with rc.

$$KL(\mathrm{P}|\mathrm{rc}^T) = \sum_{i,j} P_{i,j} \log \frac{P_{i,j}}{r_i c_j} = \sum_{i,j} P_{i,j} \log P_{i,j} - \sum_{i,j} P_{i,j} \log r_i - \sum_{i,j} P_{i,j} \log c_j$$

$r$ & $c$ :distribution $\rightarrow rc^T$ : distribution

$KL(P|rc^T) \leq \alpha$ :

= distance between P & rc should be small

= optimal solution of P(i,j) should be around rc

https://amsword.medium.com/a-simple-introduction-on-sinkhorn-distances-d01a4ef4f085

# 2. SwAV

## (2) Online Clustering

### Sinkhorn Algorithm

$$KL(\mathrm{P}|\mathrm{rc}^T) = \sum_{i,j} P_{i,j} \log \frac{P_{i,j}}{r_i c_j} = \sum_{i,j} P_{i,j} \log P_{i,j} - \sum_{i,j} P_{i,j} \log r_i - \sum_{i,j} P_{i,j} \log c_j$$

$$h(r) + h(c) - h(P) \leq \alpha$$

Interpretation of Constraint :

- **entropy of P should be large as possible!**
- non-convex problem

Solve using Lagranginan Method

**Dual Sinkhorn distance**

$$\hat{d} = \min \sum_{i,j} P_{i,j} C_{i,j} - \frac{1}{\lambda} h(P)$$

$$\sum_j P_{i,j} = r_i$$

$$\sum_i P_{i,j} = c_j$$

https://amsword.medium.com/a-simple-introduction-on-sinkhorn-distances-d01a4ef4f085

# 2. SwAV

## (2) Online Clustering

**Sinkhorn Algorithm**

$$L = \sum_{i,j} P_{i,j} C_{i,j} - \frac{1}{\lambda} h(P) + \sum_i m_i \left( \sum_j P_{i,j} - r_i \right) + \sum_j n_j \left( \sum_i P_{i,j} - c_j \right)$$

Fig. 8. Langrage form of the dual Sinkhorn distance problem.

$$\frac{\partial L}{\partial P_{i,j}} = C_{i,j} + \frac{1}{\lambda} + \frac{1}{\lambda} \log P_{i,j} + m_i + n_j = 0$$

$$P_{i,j} = e^{-\lambda m_i - 0.5} e^{-\lambda C_{i,j}} e^{-\lambda m_j - 0.5}$$

$$P_{i,j} = u_i e^{-\lambda C_{i,j}} v_j$$

$$P = \mathrm{diag(u)} \mathrm{e}^{-\lambda C} \mathrm{diag(v)}$$

Fig. 9. The derivative over P should be 0. We introduce another parameter of u and v to represent a function of m and n.

https://amsword.medium.com/a-simple-introduction-on-sinkhorn-distances-d01a4ef4f085

# 2. SwAV

## (2) Online Clustering

**Sinkhorn Algorithm**

$$\frac{\partial L}{\partial P_{i,j}} = C_{i,j} + \frac{1}{\lambda} + \frac{1}{\lambda}\log P_{i,j} + m_i + n_j = 0$$

$$P_{i,j} = e^{-\lambda m_i - 0.5} e^{-\lambda C_{i,j}} e^{-\lambda m_j - 0.5}$$

$$P_{i,j} = u_i e^{-\lambda C_{i,j}} v_j$$

$$P = \mathrm{diag(u)} \mathrm{e}^{-\lambda C} \mathrm{diag(v)}$$

Fig. 9. The derivative over P should be 0. We introduce another parameter of u and v to represent a function of m and n.

↔

$Q$ = code matrix, that connects $Z$ & $C$

$C^T Z$ = (negative) cost matrix

$$\max_{\mathbf{Q} \in \mathcal{Q}} \operatorname{Tr}\left(\mathbf{Q}^\top \mathbf{C}^\top \mathbf{Z}\right) + \varepsilon H(\mathbf{Q})$$

$$\mathcal{Q} = \left\{ \mathbf{Q} \in \mathbb{R}_+^{K \times B} \mid \mathbf{Q}\mathbf{1}_B = \frac{1}{K}\mathbf{1}_K, \mathbf{Q}^\top \mathbf{1}_K = \frac{1}{B}\mathbf{1}_B \right\}$$

$$\mathbf{Q}^* = \operatorname{Diag}(\mathbf{u}) \exp\left(\frac{\mathbf{C}^\top \mathbf{Z}}{\varepsilon}\right) \operatorname{Diag}(\mathbf{v})$$

where $\mathbf{u}$ and $\mathbf{v}$ are renormalization vectors in $\mathbb{R}^K$ and $\mathbb{R}^B$ respectively

# 2. SwAV

## (2) Online Clustering

$$\max_{\mathbf{Q}\in\mathcal{Q}} \operatorname{Tr}\left(\mathbf{Q}^\top \mathbf{C}^\top \mathbf{Z}\right) + \varepsilon H(\mathbf{Q}).$$

- $H$ : entropy function
  - $H(\mathbf{Q}) = -\sum_{ij} \mathbf{Q}_{ij} \log \mathbf{Q}_{ij}.$
- $\varepsilon$ : parameter that controls the smoothness of the mapping
  - high $\varepsilon$ : rivial solution where all samples collapse into an unique representation
  - thus, keep it low

Notation

- Feature vectors : $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_B]$
- Codes : $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_B]$
- Prototype vectors : $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_K]$

$\rightarrow$ optimize $\mathbf{Q}$ to maximize **similarity between features & prototypes**

# 2. SwAV

## (3) Multi-crop

- new **data augmentation** strategy
- mix of views with **different resolutions**
- sampling multi random crops with **2 different sizes ( standard & small )**
    - 2 standard resolution crops
    - V additional low resolution crops

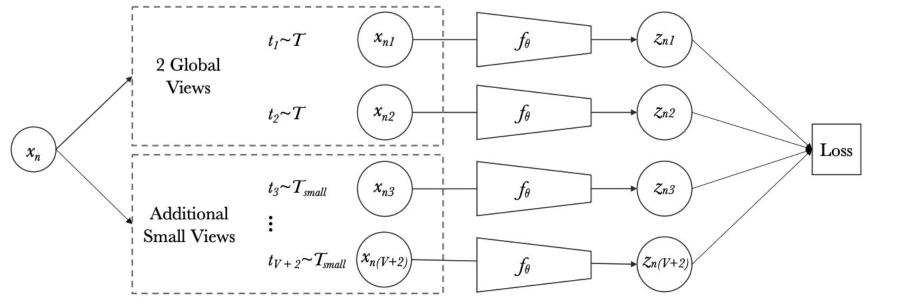



Figure 5: **Multi-crop**: the image $x_n$ is transformed into $V+2$ views: two global views and $V$ small resolution zoomed views.

$$L(\mathbf{z}_{t_1}, \mathbf{z}_{t_2}, \ldots, \mathbf{z}_{t_{V+2}}) = \sum_{i \in \{1,2\}} \sum_{v=1}^{V+2} \mathbf{1}_{v \neq i} \ell(\mathbf{z}_{t_v}, \mathbf{q}_{t_i})$$

# 3. Experiment

## (1) Evaluating the unsupervised features on ImageNet

| Method | Arch. | Param. | Top1 |
|---|---|---|---|
| Supervised | R50 | 24 | 76.5 |
| Colorization [65] | R50 | 24 | 39.6 |
| Jigsaw [46] | R50 | 24 | 45.7 |
| NPID [58] | R50 | 24 | 54.0 |
| BigBiGAN [15] | R50 | 24 | 56.6 |
| LA [68] | R50 | 24 | 58.8 |
| NPID++ [44] | R50 | 24 | 59.0 |
| MoCo [24] | R50 | 24 | 60.6 |
| SeLa [2] | R50 | 24 | 61.5 |
| PIRL [44] | R50 | 24 | 63.6 |
| CPC v2 [28] | R50 | 24 | 63.8 |
| PCL [37] | R50 | 24 | 65.9 |
| SimCLR [10] | R50 | 24 | 70.0 |
| MoCov2 [11] | R50 | 24 | 71.1 |
| SwAV | R50 | 24 | **75.3** |

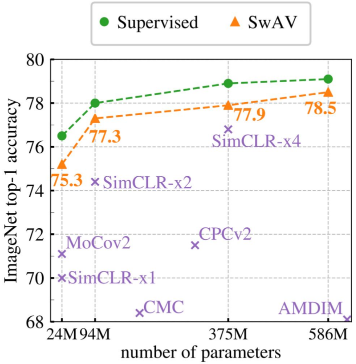



Figure 2: **Linear classification on ImageNet.** Top-1 accuracy for linear models trained on frozen features from different self-supervised methods. **(left)** Performance with a standard ResNet-50. **(right)** Performance as we multiply the width of a ResNet-50 by a factor ×2, ×4, and ×5.

Table 1: **Semi-supervised learning on ImageNet with a ResNet-50.** We finetune the model with 1% and 10% labels and report top-1 and top-5 accuracies. *: *uses RandAugment [12].*

| | | 1% labels | | 10% labels | |
|---|---|---|---|---|---|
| | Method | Top-1 | Top-5 | Top-1 | Top-5 |
| | Supervised | 25.4 | 48.4 | 56.4 | 80.4 |
| *Methods using label-propagation* | UDA [60] | - | - | 68.8* | 88.5* |
| | FixMatch [51] | - | - | **71.5*** | 89.1* |
| *Methods using self-supervision only* | PIRL [44] | 30.7 | 57.2 | 60.4 | 83.8 |
| | PCL [37] | - | 75.6 | - | 86.2 |
| | SimCLR [10] | 48.3 | 75.5 | 65.6 | 87.8 |
| | SwAV | **53.9** | **78.5** | 70.2 | **89.9** |

# 3. Experiment

## (2) Transferring unsupervised features to downstream tasks

| | Linear Classification | | | Object Detection | | |
|---|---|---|---|---|---|---|
| | Places205 | VOC07 | iNat18 | VOC07+12 (Faster R-CNN R50-C4) | COCO (Mask R-CNN R50-FPN) | COCO (DETR) |
| Supervised | 53.2 | 87.5 | 46.7 | 81.3 | 39.7 | 40.8 |
| RotNet [19] | 45.0 | 64.6 | - | - | - | |
| NPID++ [44] | 46.4 | 76.6 | 32.4 | 79.1 | - | |
| MoCo [24] | 46.9† | 79.8† | 31.5† | 81.5 | - | |
| PIRL [44] | 49.8 | 81.1 | 34.1 | 80.7 | - | |
| PCL [37] | 49.8 | 84.0 | - | - | - | |
| BoWNet [19] | 51.1 | 79.3 | - | 81.3 | - | |
| SimCLR [10] | 53.3† | 86.4† | 36.2† | - | - | |
| MoCov2 [24] | 52.9† | 87.1† | 38.9† | 82.5 | 39.8 | 42.0† |
| SwAV | **56.7** | **88.9** | **48.6** | **82.6** | **41.6** | **42.1** |

# 3. Experiment

## (3) Training with small batches

Table 3: **Training in small batch setting.** Top-1 accuracy on ImageNet with a linear classifier trained on top of frozen features from a ResNet-50. All methods are trained with a batch size of 256. We also report the number of stored features, the type of cropping used and the number of epochs.

| Method | Mom. Encoder | Stored Features | multi-crop | epoch | batch | Top-1 |
|---|---|---|---|---|---|---|
| SimCLR | | 0 | $2\times224$ | 200 | 256 | 61.9 |
| MoCov2 | ✓ | 65,536 | $2\times224$ | 200 | 256 | 67.5 |
| MoCov2 | ✓ | 65,536 | $2\times224$ | 800 | 256 | 71.1 |
| SwAV | | 3,840 | $2\times160+4\times96$ | 200 | 256 | 72.0 |
| SwAV | | 3,840 | $2\times224+6\times96$ | 200 | 256 | 72.7 |
| SwAV | | 3,840 | $2\times224+6\times96$ | 400 | 256 | **74.3** |

# 3. Experiment

## (4) Applying the multi-crop strategy to different methods

| Method | Top-1 2x224 | Top-1 2x160+4x96 | Δ |
|---|---|---|---|
| Supervised | 76.5 | 76.0 | −0.5 |
| *Contrastive-instance approaches* | | | |
| SimCLR | 68.2 | 70.6 | +2.4 |
| *Clustering-based approaches* | | | |
| SeLa-v2 | 67.2 | 71.8 | +4.6 |
| DeepCluster-v2 | 70.2 | 74.3 | +4.1 |
| SwAV | 70.1 | 74.1 | +4.0 |

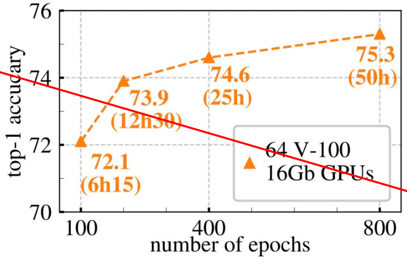
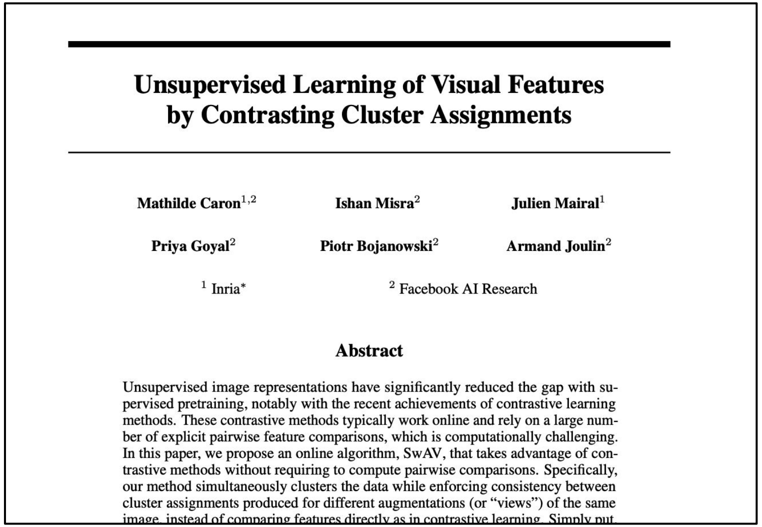



Figure 3: Top-1 accuracy on ImageNet with a linear classifier trained on top of frozen features from a ResNet-50. **(left) Comparison between clustering-based and contrastive instance methods and impact of multi-crop.** Self-supervised methods are trained for 400 epochs and supervised models for 200 epochs. **(right) Performance as a function of epochs.** We compare SwAV models trained with different number of epochs and report their running time based on our implementation.

instance. For example, in the case of 2x160+4x96 crops, we have $M = 6$ crops per instance. We call $N = B \times M$ the effective total number of crops in the batch. Overall, we minimize the following loss

$$\mathcal{L} = -\frac{1}{N}\frac{1}{M-1}\sum_{i=1}^{N}\sum_{v^+\in\{v_i^+\}} \log \frac{\exp z_i^T v^+/\tau}{\exp z_i^T v^+/\tau + \sum_{v^-\in\{v_i^-\}} \exp z_i^T v^-/\tau}. \quad (7)$$

# Papers



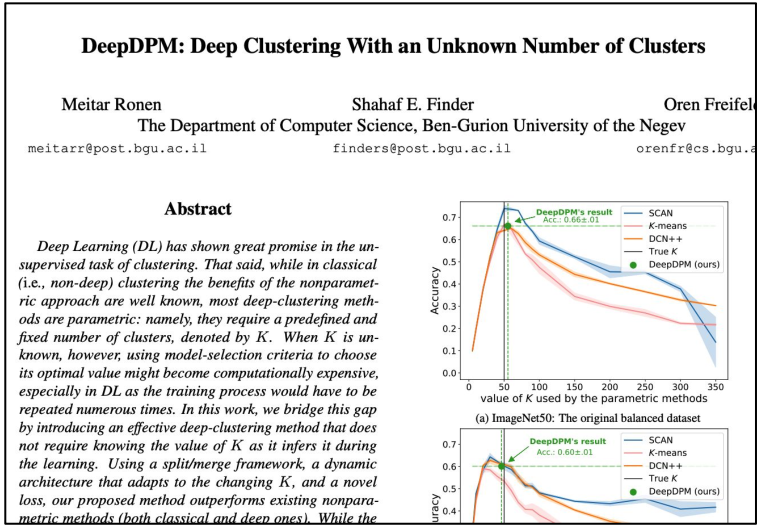
**Unsupervised Learning of Visual Features by Contrasting Cluster Assignments**

Mathilde Caron[1,2] Ishan Misra[2] Julien Mairal[1]
Priya Goyal[2] Piotr Bojanowski[2] Armand Joulin[2]
[1] Inria* [2] Facebook AI Research

**Abstract**

Unsupervised image representations have significantly reduced the gap with supervised pretraining, notably with the recent achievements of contrastive learning methods. These contrastive methods typically work online and rely on a large number of explicit pairwise feature comparisons, which is computationally challenging. In this paper, we propose an online algorithm, SwAV, that takes advantage of contrastive methods without requiring to compute pairwise comparisons. Specifically, our method simultaneously clusters the data while enforcing consistency between cluster assignments produced for different augmentations (or "views") of the same image, instead of comparing features directly as in contrastive learning. Simply put,

**SwAV (2021)**



**DeepDPM: Deep Clustering With an Unknown Number of Clusters**

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

**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*



(a) ImageNet50: The original balanced dataset

**DeepDPM (2022)**

# DeepDPM (2022)

1. DL vs Classical Clustering
2. DPGMM-based Clustering
3. DeepDPM
    a. Architecture
    b. Split & Merge
    c. Proposed Loss Function
4. Experiments

# 1. DL vs Classical Clustering

**Classical Clustering** :

- benefits from “**NON-parametric**” approach

**DL Clustering** :

- mostly “**parametric**” approach
- require a “**pre-defined # of clusters (= K)**”
- cluster “**large & high-dim**” datasets better & more efficiently

# 1. DL vs Classical Clustering

Benefits of ability to **infer K**

- (1) ***without good estimate of K***, parametric methods ***suffer in performance***
- (2) finding K with model selection -> ***computationally expensive!***

# 2. DPGMM-based Clustering

DPGMM : Dirichlet Process GMM

## (1) Notation

- $\mathcal{X} = (\boldsymbol{x}_i)_{i=1}^{N}$ : $N$ data points of $d$ dimension
- clustering task : partition $\mathcal{X}$ into $K$ disjoint groups
    - $z_i$ : cluster label of $\boldsymbol{x}_i$
- data of certain cluster : $(\boldsymbol{x}_i)_{i:z_i=k}$

# 2. DPGMM-based Clustering

## (2) DPGMM ( Dirichlet Process Gaussian Mixture Model )

- mixture with infinitely-many Gaussians
- often used, when $K$ is unknown
- $p\left(\boldsymbol{x} \mid (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k)_{k=1}^{\infty}\right) = \sum_{k=1}^{\infty} \pi_k \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right).$

Component : $\boldsymbol{\theta}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

- $\boldsymbol{\theta} = (\boldsymbol{\theta}_k)_{k=1}^{\infty}.$
- $\boldsymbol{\pi} = (\pi_k)_{k=1}^{\infty}.$
- assumed to be drawn from their own prior

# 3. DeepDPM

**DeepDPM = DL + DPM (Dirichlet Process Mixture)**

**Effective Deep-clustering** method, that ***does not require knowing # of clusters***

- (1) **Architecture**
- (2) **Split & Merge**
  - to dynamically change K
- (3) **Novel loss function**
  - for EM algorithms in mixture models

# 3. DeepDPM

## (1) Architecture

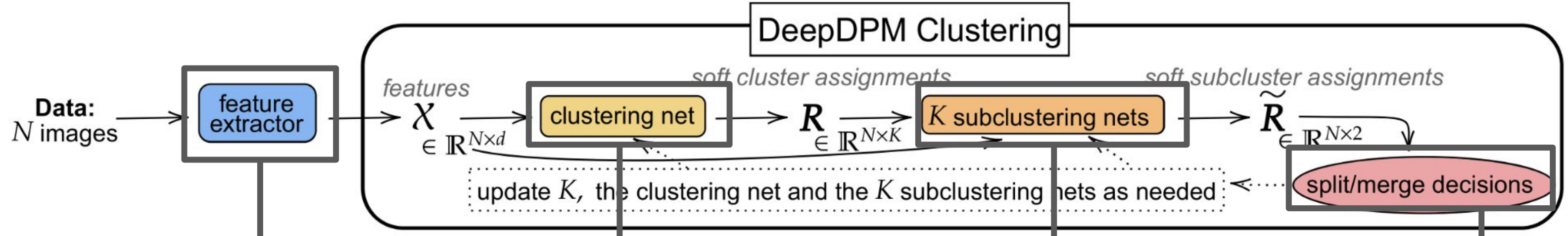



Figure 2. DeepDPM's pipeline: given features $\mathcal{X}$, the clustering net outputs cluster assignments, $\boldsymbol{R}$, while the subclustering nets generate subcluster assignments, $\widetilde{\boldsymbol{R}}$. Upon the acceptance of split/merge proposals, all those nets are updated during the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

Embedding
( use MoCo )

Cluster A, B, .., K

Cluster A-1, A-2
….
Cluster K-1, K-2

change K

# 3. DeepDPM

## (1) Architecture

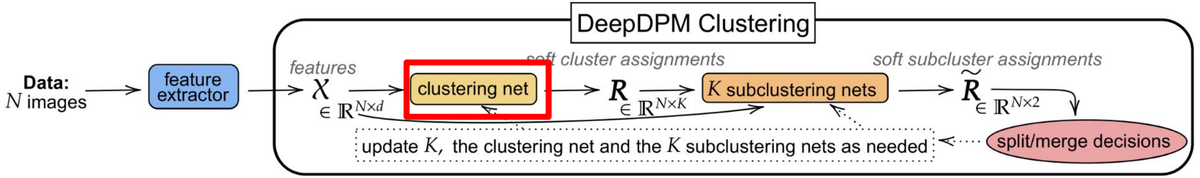



Figure 2. DeepDPM's pipeline: given features $\mathcal{X}$, the clustering net outputs cluster assignments, $\boldsymbol{R}$, while the subclustering nets generate subcluster assignments, $\widetilde{\boldsymbol{R}}$. Upon the acceptance of split/merge proposals, all those nets are updated during the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

- a) Clustering Net

$$f_{\mathrm{cl}}(\mathcal{X}) = \boldsymbol{R} = (\boldsymbol{r}_i)_{i=1}^N \quad \boldsymbol{r}_i = (r_{i,k})_{k=1}^K.$$

- for each data point $\boldsymbol{x}_i$, generate $K$ soft cluster assignments
- where $r_{i,k} \in [0,1]$ is the soft assignment ( $\sum_{k=1}^K r_{i,k} = 1$ )

Hard assignment

- from (soft) $(\boldsymbol{r}_i)_{i=1}^N$, compute (hard) $\boldsymbol{z} = (z_i)_{i=1}^N$

  ( $z_i = \arg\max_k r_{i,k}$ )

# 3. DeepDPM

## (1) Architecture

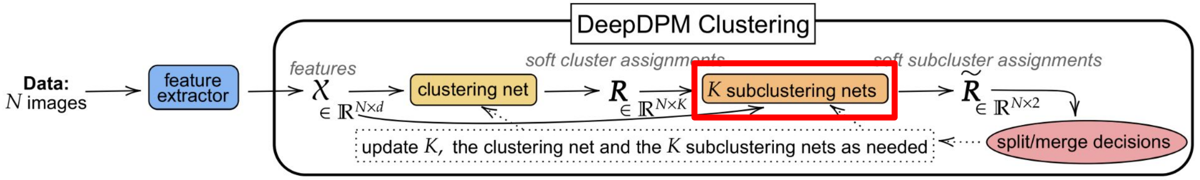



Figure 2. DeepDPM's pipeline: given features $\mathcal{X}$, the clustering net outputs cluster assignments, $\boldsymbol{R}$, while the subclustering nets generate subcluster assignments, $\widetilde{\boldsymbol{R}}$. Upon the acceptance of split/merge proposals, all those nets are updated during the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

- b) K Subclustering Net

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

- $\boldsymbol{z} = (z_i)_{i=1}^N$ is fed into $f_{\text{sub}}^k$ ( to its respective cluster )

  $\rightarrow$ generates soft subcluster assignments
- where $\tilde{r}_{i,j} \in [0,1]$ is the soft assignment of $\boldsymbol{x}_i$ to subcluster $j (j \in \{1,2\})$
  - $\tilde{r}_{i,1} + \tilde{r}_{i,2} = 1.$

Subclusters learned by $(f_{\text{sub}}^k)_{k=1}^K$ are used in split proposals.

# 3. DeepDPM

## (1) Architecture

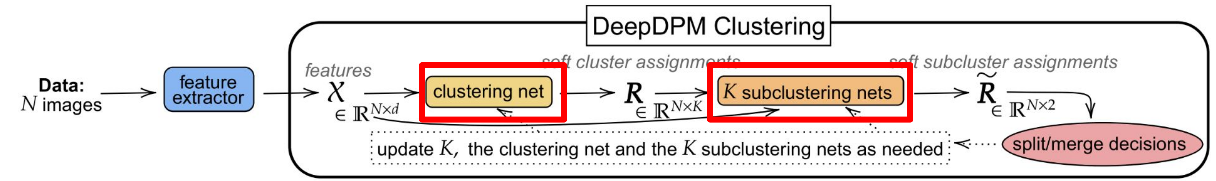



Figure 2. DeepDPM's pipeline: given features $\mathcal{X}$, the clustering net outputs cluster assignments, $\boldsymbol{R}$, while the subclustering nets generate subcluster assignments, $\widetilde{\boldsymbol{R}}$. Upon the acceptance of split/merge proposals, all those nets are updated during the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

- a) Clustering Net & b) K Subclustering Net

Each of the $K+1$ nets $(f_{\text{cl}}$ and $(f_{\text{sub}}^k)_{k=1}^K)$:

- MLP with single hidden layer
- Neurons of last layer :
  - $f_{\text{cl}}$ : $K$ neurons
  - each $f_{\text{sub}}^k$ : $2$ neurons

# 3. DeepDPM

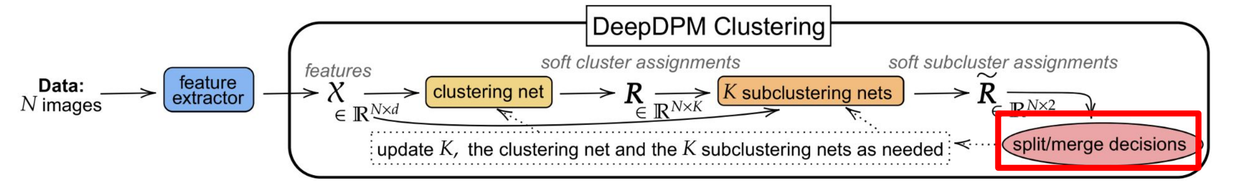
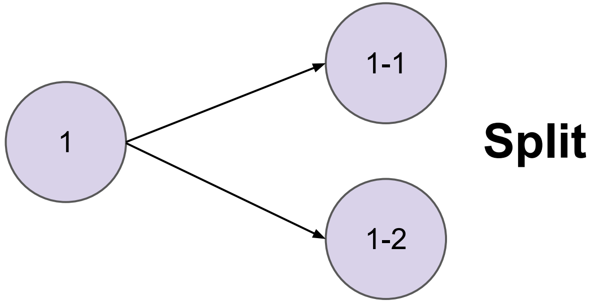
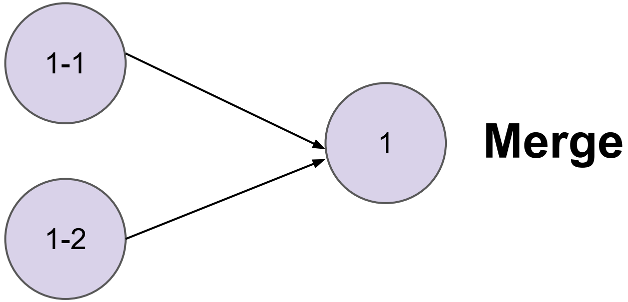



Figure 2. DeepDPM's pipeline: given features $\mathcal{X}$, the clustering net outputs cluster assignments, $\boldsymbol{R}$, while the subclustering nets generate subcluster assignments, $\widetilde{\boldsymbol{R}}$. Upon the acceptance of split/merge proposals, all those nets are updated during the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

## (2) Split & Merge

- Split : cluster + 1
- Merge : cluster -1



# 3. DeepDPM

## (2) Split & Merge

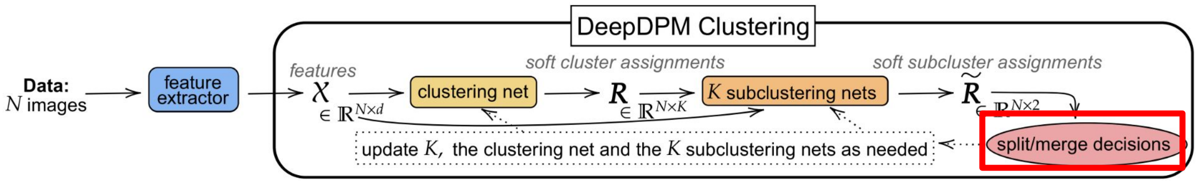



Figure 2. DeepDPM's pipeline: given features $\mathcal{X}$, the clustering net outputs cluster assignments, $\boldsymbol{R}$, while the subclustering nets generate subcluster assignments, $\widetilde{\boldsymbol{R}}$. Upon the acceptance of split/merge proposals, all those nets are updated during the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

augments latent variables with auxilairy variables

- latent variables : $(\boldsymbol{\theta}_k)_{k=1}^{\infty}$, $\boldsymbol{\pi}$, $(z_i)_{i=1}^{N}$
- auxiliary variables :
  - to each $z_i$, an additional subcluster label, $\tilde{z}_i \in \{1, 2\}$, is added.
  - to each $\boldsymbol{\theta}_k$, two subcomponents are added, $\tilde{\boldsymbol{\theta}}_{k,1}$, $\tilde{\boldsymbol{\theta}}_{k,2}$, with nonnegative weights $\tilde{\pi}_k = (\tilde{\pi}_{k,j})_{j\in\{1,2\}}$
    - where $\tilde{\pi}_{k,1} + \tilde{\pi}_{k,2} = 1$

  $\rightarrow$ 2-component GMM

# 3. DeepDPM

## (2) Split & Merge

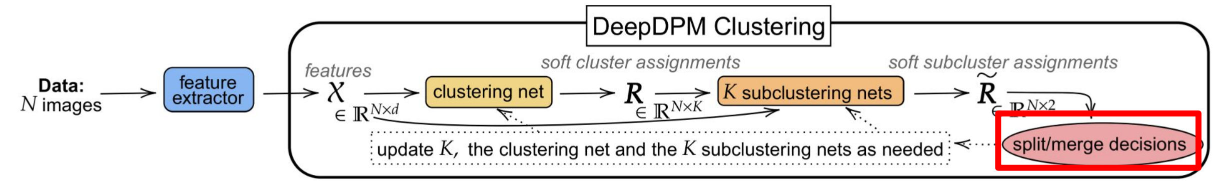



Figure 2. DeepDPM's pipeline: given features $\mathcal{X}$, the clustering net outputs cluster assignments, $\boldsymbol{R}$, while the subclustering nets generate subcluster assignments, $\widetilde{\boldsymbol{R}}$. Upon the acceptance of split/merge proposals, all those nets are updated during the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

MH-framework

- allow changing $K$ during training
- split of cluster $k$ into its subclusters is proposed
- split acceptance ratio :
  - $$H_s = \frac{\alpha\Gamma(N_{k,1})f_{\boldsymbol{x}}(\mathcal{X}_{k,1};\lambda)\Gamma(N_{k,2})f_{\boldsymbol{x}}(\mathcal{X}_{k,2};\lambda)}{\Gamma(N_k)f_{\boldsymbol{x}}(\mathcal{X}_k;\lambda)}.$$
    - $\mathcal{X}_k = (\boldsymbol{x}_i)_{i:z_i=k}$ : points in cluster $k$
    - $N_k = |\mathcal{X}_k|$ : number of points in cluster $k$
    - $f_{\boldsymbol{x}}(\cdot;\lambda)$ : marginal likelihood

interpretation : **comparing the marginal likelihood of the data, under 2 subclusters with its marginal likelihood under the cluster**

Every few epochs, ***propose either SPLITS or MERGES***

# 3. DeepDPM

## (2) Split & Merge

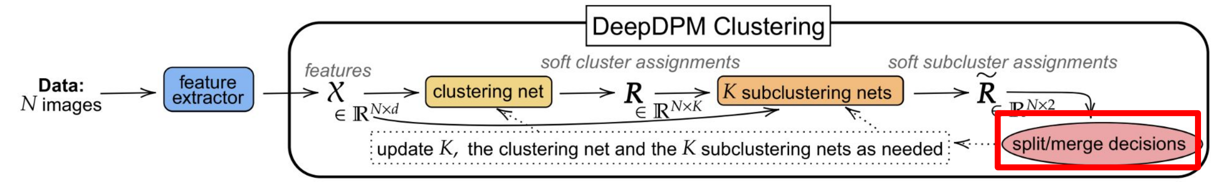



Figure 2. DeepDPM's pipeline: given features $\mathcal{X}$, the clustering net outputs cluster assignments, $\boldsymbol{R}$, while the subclustering nets generate subcluster assignments, $\widetilde{\boldsymbol{R}}$. Upon the acceptance of split/merge proposals, all those nets are updated during the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

### a) Split

propose to split each of the clusters into **2 subclusters**

- split probability = $\min(1, H_\mathrm{s})$

IF ACCEPTED ( = SPLIT ) for cluster $k$...

- (Clustering Net) $k$-th unit of last layer is **duplicated**
  - initialize the parameters of 2 new clusters, with **parametes of SUBcluster nets**
    - $$\begin{aligned} \boldsymbol{\mu}_{k_1} \leftarrow \widetilde{\boldsymbol{\mu}}_{k,1}, \quad \boldsymbol{\Sigma}_{k_1} \leftarrow \widetilde{\boldsymbol{\Sigma}}_{k,1}, \quad \pi_{k_1} \leftarrow \pi_k \times \widetilde{\boldsymbol{\pi}}_{k,1} \\ \boldsymbol{\mu}_{k_2} \leftarrow \widetilde{\boldsymbol{\mu}}_{k,2}, \quad \boldsymbol{\Sigma}_{k_2} \leftarrow \widetilde{\boldsymbol{\Sigma}}_{k,2}, \quad \pi_{k_2} \leftarrow \pi_k \times \widetilde{\boldsymbol{\pi}}_{k,2} \end{aligned}.$$
  - $k_1$ and $k_2$ : indices of the new clusters

# 3. DeepDPM

## (2) Split & Merge

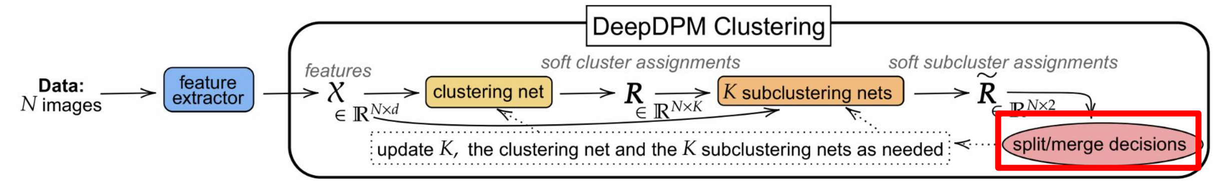
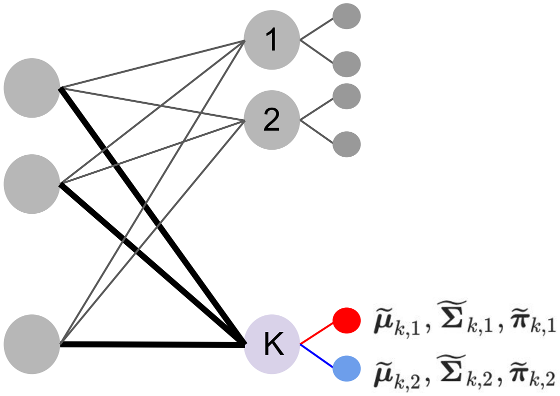



Figure 2. DeepDPM's pipeline: given features $\mathcal{X}$, the clustering net outputs cluster assignments, $\boldsymbol{R}$, while the subclustering nets generate subcluster assignments, $\widetilde{\boldsymbol{R}}$. Upon the acceptance of split/merge proposals, all those nets are updated during the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

### a) Split



# 3. DeepDPM

## (2) Split & Merge

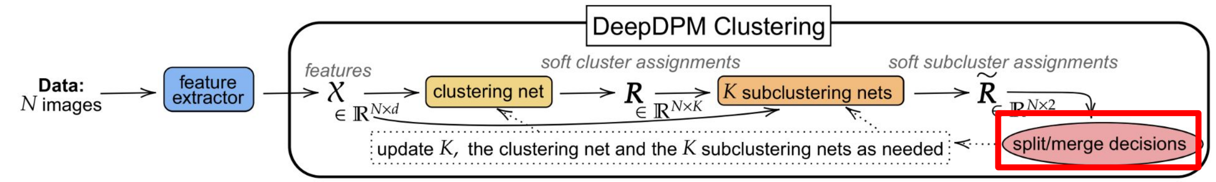
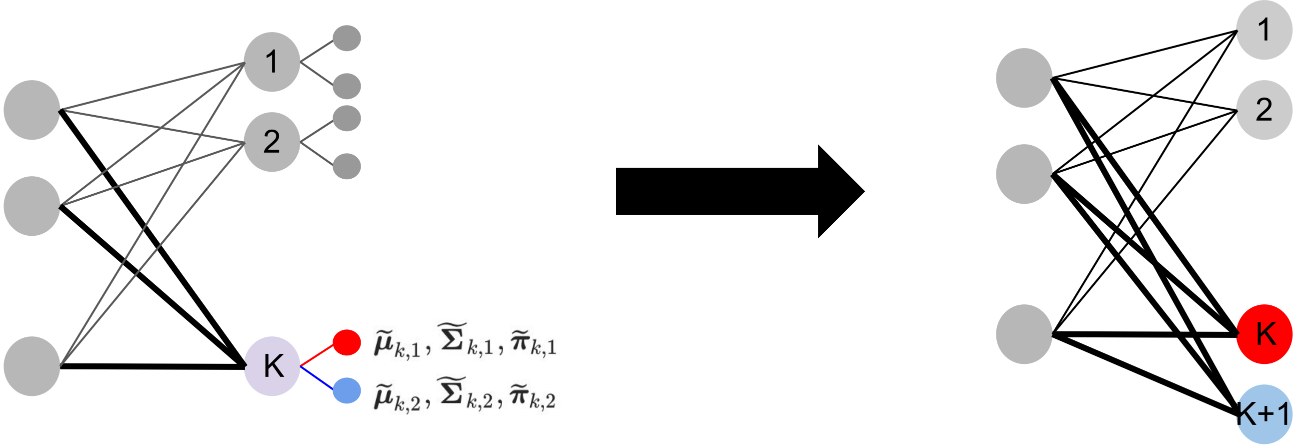



Figure 2. DeepDPM's pipeline: given features $\mathcal{X}$, the clustering net outputs cluster assignments, $\boldsymbol{R}$, while the subclustering nets generate subcluster assignments, $\widetilde{\boldsymbol{R}}$. Upon the acceptance of split/merge proposals, all those nets are updated during the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

**a) Split**




$\widetilde{\boldsymbol{\mu}}_{k,1}, \widetilde{\boldsymbol{\Sigma}}_{k,1}, \widetilde{\boldsymbol{\pi}}_{k,1}$

$\widetilde{\boldsymbol{\mu}}_{k,2}, \widetilde{\boldsymbol{\Sigma}}_{k,2}, \widetilde{\boldsymbol{\pi}}_{k,2}$

$\widetilde{\boldsymbol{\mu}}_{k,1}, \widetilde{\boldsymbol{\Sigma}}_{k,1}, \pi_k \times \widetilde{\boldsymbol{\pi}}_{k,1}$

$\widetilde{\boldsymbol{\mu}}_{k,2}, \widetilde{\boldsymbol{\Sigma}}_{k,2}, \pi_k \times \widetilde{\boldsymbol{\pi}}_{k,2}$

# 3. DeepDPM

## (2) Split & Merge

### b) Merge

Splits vs Merge

- Splits : can be done in **parallel**
- Merge : cannot ~

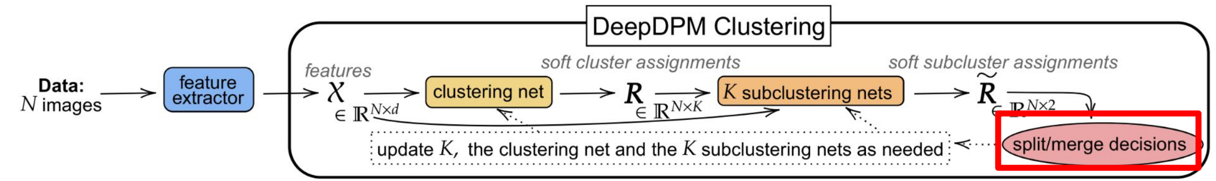



Figure 2. DeepDPM's pipeline: given features $\mathcal{X}$, the clustering net outputs cluster assignments, $\boldsymbol{R}$, while the subclustering nets generate subcluster assignments, $\widetilde{\boldsymbol{R}}$. Upon the acceptance of split/merge proposals, all those nets are updated during the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

To avoid sequentially considering all possible merges...

→ merges of each cluster **with only its 3 nearest neighbors**

# 3. DeepDPM

## (2) Split & Merge

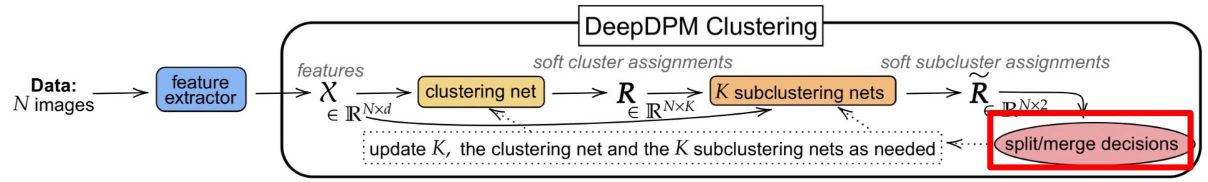



Figure 2. DeepDPM's pipeline: given features $\mathcal{X}$, the clustering net outputs cluster assignments, $\boldsymbol{R}$, while the subclustering nets generate subcluster assignments, $\widetilde{\boldsymbol{R}}$. Upon the acceptance of split/merge proposals, all those nets are updated during the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

### b) Merge

Splits vs Merge

- Splits : can be done in **parallel**
- Merge : cannot ~

→ To avoid sequentially considering all possible merges...

→ merges of each cluster **with only its 3 nearest neighbors**

Merge probability : $H_{\mathrm{m}} = 1/H_{\mathrm{s}}$

IF ACCEPTED ( = MERGE ) ...

- 2 clusters are merged
- **new subcluster network** of the merged clusters is made
- one of the 2 clusters' weight (connected to the last layer) is removed from $f_{cl}$

# 3. DeepDPM

## (2) Split & Merge

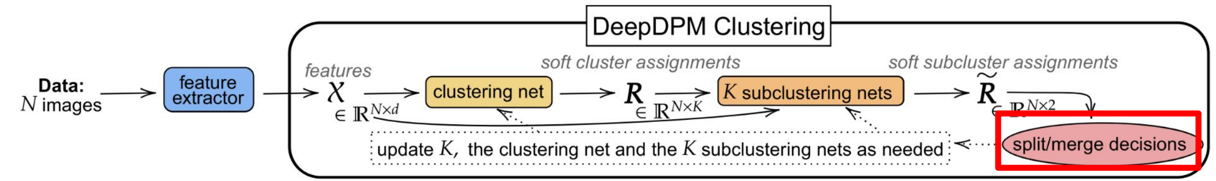
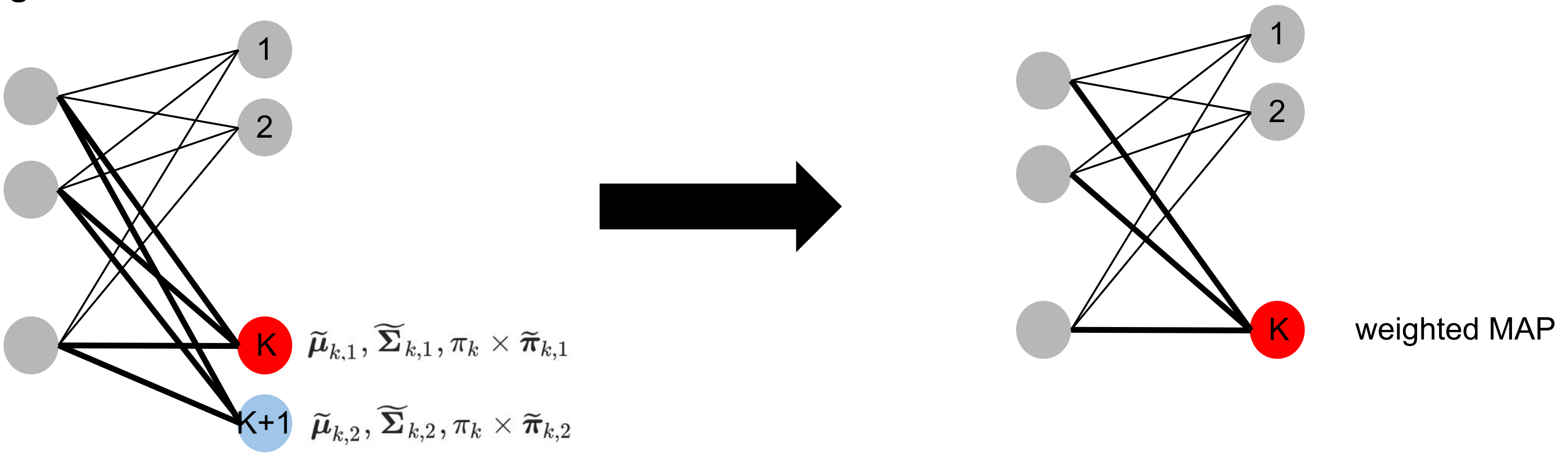



Figure 2. DeepDPM's pipeline: given features $\mathcal{X}$, the clustering net outputs cluster assignments, $\boldsymbol{R}$, while the subclustering nets generate subcluster assignments, $\widetilde{\boldsymbol{R}}$. Upon the acceptance of split/merge proposals, all those nets are updated during the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

**b) Merge**



# 3. DeepDPM

## (3) Novel loss function

motivated by EM algorithm in Bayesian GMM

( iterative procedure )

- **[ E step ] assign cluster**
- **[ M step ] update cluster parameter**

# 3. DeepDPM

## (3) Novel loss function

### [ E step ] assign cluster

- For each $\boldsymbol{x}_i$ and each $k \in \{1, \ldots, K\}$, compute E-step probabilities $\boldsymbol{r}_i^{\mathrm{E}} = \left(r_{i,k}^{\mathrm{E}}\right)_{k=1}^{K}$
  - $r_{i,k}^{\mathrm{E}} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k'=1}^{K} \pi_{k'} \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})}$ $k \in \{1, \ldots, K\}$. → **soft cluster assignment**
- computed using $(\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_{k=1}^{K}$ from previous epochs

encourage $f_{\mathrm{cl}}$ to generate similar soft assignments using the following new loss:

- $\mathcal{L}_{\mathrm{cl}} = \sum_{i=1}^{N} \mathrm{KL}\left(\boldsymbol{r}_i \,||\, \boldsymbol{r}_i^{\mathrm{E}}\right)$.

# 3. DeepDPM

## (3) Novel loss function

### [ M step ] update cluster parameter

- uses the weighted versions of the MAP estimates of $(\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_{k=1}^{K}$ ,
  where the weights are...
  - $r_{i,k}^{\mathrm{E}}$ (X)
  - $r_{i,k}$ (O) $\rightarrow$ output of $f_{cl}$

for $(f_{\text{sub}}^{k})_{k=1}^{K}$ ... calculate Isotropic Loss :

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

- where $N_k = | \mathcal{X}_k |$
- $\tilde{\boldsymbol{\mu}}_{k,j}$ : mean of subcluster $j$ of cluster $k$

# 4. Experiments

## (1) 3 Common Metrics ( Higher = Better )

1. Clustering Accuracy ( **ACC** )

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

$$H(X) = -\sum_x P(x) log P(x)$$

2. Normalized Mutual Information ( **NMI** )

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

$$\begin{aligned} I(X;Y) &= \sum_x \sum_y P(X,Y) \log \frac{P(X,Y)}{P(X)P(Y)} \\ &= \sum_x P(X) H(Y \mid X) - \sum_y P(Y) \log P(Y) \\ &= -H(Y \mid X) + H(Y) \\ &= H(Y) - H(Y \mid X) \end{aligned}$$

3. Adjusted Rand Index ( **ARI** )

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

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

$a_k$ : sum of row $k$ in contingency table

$b_l$ : sum of column $l$ in contingency table

$c_{kl} = |y_k \cap z_k|$

TP : ( same cluster & same label ), TN : ( different cluster & different label )
FP : ( same cluster & different label ), FN : ( different cluster & same label )

# 4. Experiments

## (2) Comparison with classical methods

( Parametric : K-means, GMM // Non-parametric : DBSCAN, moVB, DPM sampler )

| | 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.

# 4. Experiments

## (2) Comparison with classical methods

among the nonparametric methods, DeepDPM's inferred  is the closest to the GT

| 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 $K = 10$.

# 4. Experiments

## (3) Comparison with deep non-parametric methods

| | 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.

# 4. Experiments

## (4) Clustering the entire ImageNet dataset

initialized with K=200, and converged to K=707 … ( GT : K = 1000 )

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

# 4. Experiments

## (5) Class Imbalance

| 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.

Thank You !