DNB: A Joint Learning Framework for Deep Bayesian Nonparametric Clustering

Zeya Wang, Yang Ni, Baoyu Jing, Deqing Wang, Hao Zhang, and Eric Xing, *Fellow, IEEE*

***Abstract*— Clustering algorithms based on deep neural networks have been widely studied for image analysis. Most existing methods require partial knowledge of the true labels, namely, the number of clusters, which is usually not available in practice. In this article, we propose a Bayesian nonparametric framework, deep nonparametric Bayes (DNB), for jointly learning image clusters and deep representations in a doubly unsupervised manner. In doubly unsupervised learning, we are dealing with the problem of "unknown unknowns," where we estimate not only the unknown image labels but also the unknown number of labels as well. The proposed algorithm alternates between generating a potentially unbounded number of clusters in the forward pass and learning the deep networks in the backward pass. With the help of the Dirichlet process mixtures, the proposed method is able to partition the latent representations space without specifying the number of clusters *a priori*. An important feature of this work is that all the estimation is realized with an end-to-end solution, which is very different from the methods that rely on *post hoc* analysis to select the number of clusters. Another key idea in this article is to provide a principled solution to the problem of "trivial solution" for deep clustering, which has not been much studied in the current literature. With extensive experiments on benchmark datasets, we show that our doubly unsupervised method achieves good clustering performance and outperforms many other unsupervised image clustering methods.**

***Index Terms*— Bayesian nonparametrics (BNPs), convolutional neural network (CNN), image clustering, joint learning, regularization.**

I. Introduction

RECENT advancement in deep learning has brought breakthroughs in the field of supervised learning. In particular, convolutional neural networks (CNNs) have been extensively used in computer vision to predict image labels and extract convolutional features. Although the existing state-of-the-art algorithms have achieved very high accuracy, supervised learning requires a great amount of annotated data, which is limited by the availability of labels and the huge cost of collecting and annotating the images. Unsupervised learning, particularly clustering, is useful to automatically create labels and extract features from unlabeled data.

Even though clustering methods have been broadly studied in general machine learning literature [1], [2], the majority of them are not scalable to large-scale datasets. Thus, it has brought to researchers' attention for the development of clustering methods that are based on deep neural networks, which is still relatively immature for image analysis. Convolutional autoencoders, which are able to learn the image representations through minimizing a reconstruction loss, have been employed frequently for clustering images [3], [4]. Directly using the embedding data learned from autoencoders for clustering is deemed not efficient enough, given that they consist of two stages for extracting the low-dimensional representation and learning the clustering behaviors, and the features that are optimal in reconstructing the raw images are not necessarily the most discriminative for clustering. Despite a growing number of studies on jointly doing these two tasks with autoencoder [5], [6], the symmetry architecture makes the computational cost for obtaining data embeddings from an autoencoder increasing quickly with the network depth, which limits its application to large-scale tasks in practice [7].

Recent progress has been made to jointly learn the image clusters and convolutional features with CNNs [5], [8], [9], which shows promising performance for recovering the true labels even compared with supervised learning methods. However, a big assumption of these joint methods is the knowledge of the number of clusters. This assumption is not only unrealistic in many applications, but it also limits the clustering algorithm from being self-innovative (i.e., finding new clusters). For instance, in cancer research, scientists use medical images to cluster heterogeneous patient population into homogeneous subpopulations [10], where the number of such subpopulations is unknown and estimating it is of high scientific value by itself as it measures the intertumor heterogeneity for a given cancer type. Determining the number of clusters during clustering is also an attractive property for clustering survey images with robots as it could allow truly autonomous sensor data abstraction and incorporation of new information [11]. Another important application is clustering high-dimensional astronomical images, while the number of clusters is hard to be acquired [12]. For these examples, without prior information and guidelines, it is computationally expensive or even impossible to select the number of clusters as a hyperparameter, which requires running the clustering

Manuscript received February 7, 2020; revised September 19, 2020, January 24, 2021, and May 5, 2021; accepted May 24, 2021. The work of Yang Ni was supported in part by the National Science Foundation (NSF) under Grant DMS-1918851. *(Corresponding authors: Zeya Wang; Yang Ni.)*

Zeya Wang is with Petuum, Inc., Pittsburgh, PA 15222 USA (e-mail: zw17.rice@gmail.com).

Yang Ni is with the Department of Statistics, Texas A&M University, College Station, TX 77843 USA (e-mail: yni@stat.tamu.edu).

Baoyu Jing is with the Department of Computer Science, University of Illinois at Urbana–Champaign, Champaign, IL 61820 USA.

Deqing Wang is with the School of Computer Science, Beihang University, Beijing 100083, China.

Hao Zhang and Eric Xing are with Petuum, Inc., Pittsburgh, PA 15222 USA.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2021.3085891.

Digital Object Identifier 10.1109/TNNLS.2021.3085891

algorithm many times. In this article, to highlight the distinction from the general deep clustering methods, we refer to the task of learning both cluster labels and the number of clusters as to "doubly unsupervised" learning. Doubly unsupervised learning is significantly harder than unsupervised learning, which is given in the following.

1) In unsupervised learning, the number of ways to partition a set of n images into k nonempty subsets is the Stirling number of the second kind $S(n, k)$.
2) In doubly unsupervised learning, the number of ways to partition a set of n images into an unknown number of nonempty subsets is the Bell number $B_n = \sum_{k=0}^{n} S(n, k)$, which is substantially larger than the Stirling number even for a moderate number of images.

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a collection of n images with unknown labels $Y = \{y_1, y_2, \ldots, y_n\}$. Each label $y_i \in \{1, 2, \ldots, K\}$ can take an unknown number K of possible integers. We aim to find a good mapping from the images X to their data embeddings, namely, the network parameterization, together with Y. Let θ denote the parameters of the deep clustering model and L denote a loss function. We formulate a doubly unsupervised problem for the joint learning of deep representations and image clusters as

$$\arg\min_{Y,K,\theta} L(Y, K, \theta|X). \tag{1}$$

It is not straightforward to extend the existing deep clustering algorithms to "doubly unsupervised" tasks. In this article, we propose a novel deep nonparametric Bayes (DNB) clustering algorithm with a recursive framework that takes advantage of both deep neural networks and Bayesian nonparametric (BNP) models; the latter elegantly learns the number of clusters in an automated fashion. BNP models are a class of flexible statistical models that are widely used in both supervised and unsupervised learning. Specifically to this work, our framework is based on a Dirichlet process mixture (DPM) model that allows for a potentially unbounded number of clusters (as sample size approaches infinity). Due to this limiting behavior of DPM, we do not need to prespecify the number of clusters *a priori*, and the model is able to learn it *a posteriori*. In addition, existing clustering deep neural networks (CDNNs), especially those that deploy alternating procedure, are susceptible to the problem of so-called "trivial solutions" [9], [13], where the projected latent features tightly collapsed around the cluster centers. Such a case will lead to undesirable clustering behavior even though the loss function may keep decreasing. For example, empty clusters can be obtained for the deep clustering methods with a specified number of clusters. For the methods with an alternating procedure between training the deep network and learning the unsupervised classifier, the parameterization of the network leading to an embedded subspace that assigns subjects to a few clusters will be reinforced to only discriminate between these clusters [9]. The consequence is even more serious for doubly unsupervised learning that lacks a constraint of the cluster number, where all the embedded data will be collapsed into a singular point. To address this problem, we propose a simple effective solution based on the geometric interpretation of matrix determinant, which can be potentially used in other deep clustering algorithms as well.

The major contributions of our work are as follows.

1) We formulate a doubly unsupervised problem for joint learning of deep representations and image clusters.
2) We develop a novel end-to-end training framework for the formulated doubly unsupervised problem, which can jointly learn the image clusters and deep representations without knowing the number of clusters.
3) We present an effective solution to the common problem of "trivial solution" through regularization.
4) We perform extensive experiments and ablation studies, benchmarking against alternative deep clustering methods with the optimal number of clusters selected using a couple of common clustering validity indices, to show the effectiveness and importance of the proposed framework.

II. Related Work

Clustering partitions the entities into latent groups/clusters in an unsupervised manner, with the aim of creating homogeneous groups such that observations in the same cluster are more similar to each other than to those in other clusters. It is an essential problem that has been extensively studied in machine learning [1]. A good number of classical clustering algorithms have been proposed, e.g., K-means, hierarchical clustering, and finite mixture models [1], [14]. For high-dimensional data, many clustering algorithms suffer from the curse of dimensionality, which usually requires dimension reduction to eliminate the effects of attributes that are irrelevant to the cluster structure. Yamamoto and Hwang [15], Allab *et al.* [16], Labiod and Nadif [17], and Allab *et al.* [18] proposed to embed the data on a low-dimensional space and cluster the embeddings. Instead of sequentially performing data embedding and subspace clustering, a more efficient and effective strategy is to jointly learn the embedding and clustering. Reduced k-means analysis and factorial k-means analysis are studied for such joint learning [15]. Principal component analysis (PCA), which is one of the most common tools for dimension reduction, has been unified with semi-nonnegative matrix factorization (NMF) to give more interpretable solutions while getting good clustering results [16]. Recently, spectral data embedding has also been thoroughly studied to learn the low-dimensional representation that is more suited for identifying the cluster structures [17], [18].

Similar to the data embedding technique used for subspace clustering, deep clustering has been recently developed, which projects high-dimensional data onto low-dimensional deep representation space using deep neural networks and then partitions the deep representations to create cluster labels. The optimization objective of deep clustering methods is typically minimizing a clustering loss through the deep representations generated from a deep neural network. Based on the way that a neural network determines the clustering loss, deep clustering methods can be grouped to autoencoder-based, generative adversarial-based network, and CDNN-based approaches [7].

Autoencoder is a popular structure for reconstruction tasks, which has been widely used for learning embedded space. Autoencoder consists of an encoder and a decoder, where each of them can be a fully connected neural network or a CNN, and the architecture of the decoder is usually a mirrored version of the encoder. The encoder is responsible for compressing the input data to an embedded space, and the decoder reconstructs the input data given the data embeddings. For the autoencoder-based methods, the embedded data from the encoder component are used for performing cluster analysis [5], [6], [19]. Recently, there have been a great number of methods that train/fine-tune the data embedding from autoencoders and identify cluster structures with common learning approaches, e.g., k-means [6] and Gaussian mixture models (GMMs) [20], under a unified learning framework. A generative adversarial network (GAN) [68] has also gained popularity for deep clustering in recent years. GAN builds a min-max game by alternating between training a generative network and training a discriminative network, where the generative network projects samples from a prior distribution to the data space and the discriminative network discriminates whether an input is from real data or generated from a prior distribution. Based on the underlying idea of the GAN, several models that apply the adversarial autoencoders [21], [22] or generalize GAN to multiple classes [23] have been proposed for deep clustering. However, similar to the GAN, these algorithms have the same problems of weak convergence and mode collapse [7]. In order to have a simpler training procedure and enable joint learning of deep representations and image clusters, deep neural network-based algorithms have been lately proposed to build an end-to-end clustering pipeline that improves the model scalability by directly minimizing a clustering loss on the top of a network [8], [9]. These methods only require a clustering loss and involve an iterative procedure for jointly updating the network and estimating the cluster labels. A well-designed clustering loss, as well as the training framework, can be easily adapted to any kind of network, e.g., a fully connected network, a CNN, and even a deep belief network. In addition to having a unified training pipeline, deep neural network-based methods can also avoid adding a decoder that is required by the autoencoder-based models. This difference makes it easier for deep neural network-based methods to be extended to large-scale tasks in practice. Especially considering many state-of-the-art CNN architectures developed recently, deep neural network-based methods enable a wider application to the large-scale dataset [24]. For example, DeepCluster, an end-to-end method that jointly updates network parameters and image clusters, has been successfully applied on large-scale datasets like ImageNet [25] for learning visual features [9].

The unsupervised learning algorithms typically require prior knowledge of the number of clusters, which potentially limits their practical values. Several clustering validity indices, such as the Silhouette coefficient [26], Dunn index [27], and some others [28]–[33], have been designed to compare the quality of clusters for a specific dataset and to determine the optimal number of clusters. These indices can be combined with existing clustering algorithms for the doubly supervised tasks. However, this approach often incurs a high computation cost because of repeatedly refitting the clustering model with different numbers of clusters. Density-based clustering approaches, e.g., DBSCAN [34], do not require the specification of the number of clusters but usually have some additional sensitive tuning hyperparameters. BNP clustering methods have become one of the primary choices for solving such a problem given their advantage of automatically identifying the number of clusters [35]. These methods offer a wide range of flexible alternatives, including DPM [36], [37], Pitman–Yor process mixtures [38], and normalized generalized Gamma process mixtures [39]. BNP mixtures can approximate any probability density arbitrarily well and neatly resolves the issue of finding the number of clusters. In addition, the rapid development of fast computation algorithms, such as variational Bayes (VB) [40] and consensus Monte Carlo [41], [42], allows BNP to scale up to larger datasets. For those reasons, BNP has gained great popularity in the machine learning community for doubly unsupervised learning tasks.

The recent achievement in deep clustering has successfully solved the challenge of clustering images with deep neural networks, which also encourages the development of a method to automatically determine the number of clusters with deep clustering. Only few attempts has been made to perform deep clustering assuming the number of clusters to be unknown [43], which nevertheless requires pretraining the deep network for clustering. A straightforward way to combine the general clustering algorithms that estimate the number of clusters with deep networks is to implement them directly on the deep representations generated from a well-trained network, which can be achieved through pretraining the network or a reconstruction task with an autoencoder. However, it comes with a two-stage training procedure and cannot jointly learn the deep representations, image labels, and the number of clusters. Also, network pretraining requires a lot of annotated images and sometimes needs to account for a cross-domain effect; the computational cost of learning deep representations from an autoencoder can increase greatly with the network depth, which limits its scalability to large-scale tasks [7]. In this article, we will develop a novel deep clustering method that jointly estimates the number of clusters, deep representations, and clustering labels.

III. Method

We first provide a high-level picture of the proposed method and fix our notations. Let $X = \{x_1, \ldots, x_n\}$ denote a set of unlabeled n images. Each image x_i has dimension $d = H \times W \times C$, which represents the image height, width, and the number of channels, respectively. Let $z_i = f_\theta(x_i)$ be a function that maps high-dimensional images to a much lower dimensional space. In our work, $f_\theta(\cdot)$ is a CNN with convolutional layers and fully connected layers, parameterized by θ. Let $y_i \in \{1, \ldots, K\}$ denote the unknown cluster label for image i and let $Y = \{y_1, \ldots, y_n\}$. Later, we will provide a clustering rule that minimizes a loss function L.

Clustering the raw images X is challenging due to the extremely large value of d. For that, deep clustering algorithms have been recently developed, which assumes that K is known

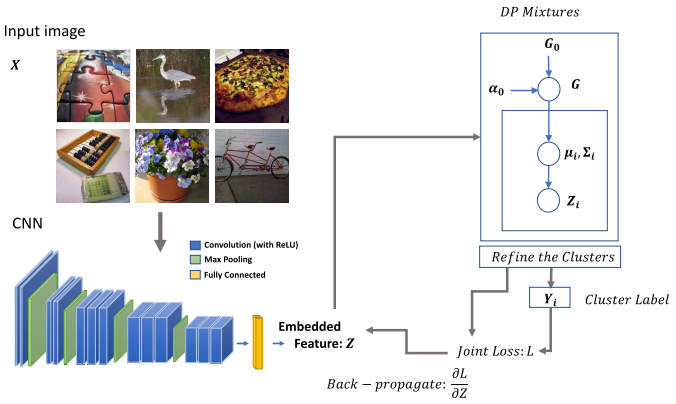


Fig. 1. Illustration of the proposed joint learning framework for deep BNP clustering.

and consider the optimization problem $\min_{Y,\theta} L(Y, \theta|X, K)$. We consider the case where K is unknown with the optimization problem given in (1). It is difficult to minimize the loss function with respect to all the parameters simultaneously due to the intricate relationship between the network parameters and clustering assignments. In addressing this challenge, we propose the DNB clustering algorithm.

A. DNB Clustering Algorithm

In order to enable the deep network to be jointly trained through backpropagation from a clustering loss, we provide an iterative optimization scheme alternating between a forward step and a backward step: first, clustering the embedded data output from the network, which creates pseudo labels and estimates cluster-specific parameters, and second, learning deep representations as well as updating the network parameters through minimizing a joint loss that we design given the pseudo labels and the associated distributions. Note that in the first step, we also learn the number of clusters. We additionally provide an extra step in the forward pass for refining the generated clusters. The flowchart of DNB is shown in Fig. 1. As we mentioned earlier, most clustering models that require a prespecified number of clusters are not suitable for doubly unsupervised learning tasks. We build our forward step based on a BNP model, a popular Bayesian inference method that is useful for automatically determining the number of clusters.

1) DPMs and Clustering: In our forward step, the deep representations $Z = \{z_1, \ldots, z_n\}$ of X are generated from the previous loop. We will focus on one particular BNP model, the DPMs because of its computational simplicity, large support, and well-understood theoretical properties; extension to other BNP mixtures, such as Pitman–Yor process mixtures and normalized generalized gamma process mixtures, is straightforward. Dirichlet processes (DPs) are a family of stochastic processes whose realizations are probability distributions. Importantly, the distributions drawn from the DP are discrete with probability one. Used as the prior distribution in Bayesian inference, the discreteness induces ties among the parameters, which in turn naturally forms clusters. DPM can be formulated as a Bayesian hierarchical model

$$z_i|\eta_i \sim p(z_i|\eta_i), \quad \eta_i|G \sim G, \ G \sim \mathrm{DP}(\alpha, G_0) \tag{2}$$

where the likelihood $p(\cdot|\cdot)$ is some known probability distribution indexed by the parameter η_i and η_i has a DP-distributed prior distribution G. DP has two parameters, the concentration parameter α and the base measure G_0, which is usually chosen to be conjugate with respect to the likelihood for efficient computation. The discreteness of G gives rise to ties among η's and the cluster labels are determined by the ties, i.e., $y_i = y_j$ if $\eta_i = \eta_j$. The concentration parameter α influences the prior number of clusters. Its value can be chosen via either an empirical Bayes approach [44] or imposing a gamma hyperprior [45]. However, we do not pursue this direction because, in our experiments, we find that its value does not significantly affect the posterior distribution.

Marginalizing out G, the hierarchical model in (2) can be equivalently expressed as an infinite mixture model

$$\begin{aligned} z_i|y_i = k \sim p\left(z_i|\eta_k^\star\right), \quad y_i|\pi \sim \mathrm{categ}(\pi), \ \pi = (\pi_1, \pi_2, \ldots) \\ \pi_k = \beta_k \prod_{l=1}^{k-1}(1-\beta_l), \quad \beta_k \sim \mathrm{beta}(1, \alpha), \ \eta_k^\star \sim G_0 \end{aligned} \tag{3}$$

where $\eta_k^\star$ is the cluster-specific parameter, π is a vector of probabilities for cluster $k = 1, 2, \ldots$, and beta and categ denote beta and categorical distributions, respectively. This construction is also known as the stick-breaking representation, which is crucial to the derivation of the VB algorithm for DPM. It is clear from this representation that clustering with DPM does not require the prespecification of the number of clusters as it can automatically grow as the sample size increases. Let $N(\cdot|a, A)$ denote a normal density with mean a and covariance A and $W(\cdot|c, D)$ denote a Wishart density with degrees of freedom c and scale matrix D (i.e., the expectation is cD). We use a normal likelihood $f(z_i|\eta_k^\star) = N(z_i|\mu_k, \Lambda_k^{-1})$ with a conjugate normal-Wishart base distribution $G_0 = N(\mu_k|\zeta, \Lambda_k^{-1}/\tau) \times W(\Lambda_k|b, \Omega^{-1})$. Even just with normal likelihood, DPM can approximate any distribution arbitrarily well.

2) Approximate Posterior Inference via VB : The DPM is indexed by four sets of parameters $\{\mu_k\}_{k=1}^\infty$, $\{\Lambda_k\}_{k=1}^\infty$, $\{\beta_k\}_{k=1}^\infty$, and $\{y_i\}_{i=1}^n$. The posterior distribution of these parameters cannot be computed directly. We use VB to approximate the posterior distribution. VB aims to minimize the Kullback–Leibler (KL) divergence between the variational distribution and the posterior distribution. We assume a fully factorized variational distribution

$$\begin{aligned} q\left(\{\mu_k\}_{k=1}^{K^*}, \{\Lambda_k\}_{k=1}^{K^*}, \{\beta_k\}_{k=1}^{K^*-1}, \{y_i\}_{i=1}^n\right) \\ = \prod_{k=1}^{K^*} q(\mu_k) q(\Lambda_k) \prod_{k=1}^{K^*-1} q(\beta_k) \prod_{i=1}^{n} q(y_i) \end{aligned} \tag{4}$$

where $q(\mu_k) = N(\mu_k|m_k, \Psi_k)$, $q(\Lambda_k) = W(\Lambda_k|c_k, D_k^{-1})$, $q(\beta_k) = \mathrm{beta}(\beta_k|\gamma_{k1}, \gamma_{k2})$, and $q(y_i) = \mathrm{categ}(\phi_i)$. The stick-breaking process is truncated at level K^* so that $q(\beta_{K^*} = 1) = 1$. We remark that K^* is not a prespecified number of clusters as in other clustering algorithms. Rather, K^* is simply an upper bound for the number of clusters here, and hence, any large enough K^* (e.g., 100) leads to the same results. Since both the full conditional distribution and the variational distribution are exponential family distributions,

minimizing KL divergence with respect to the variational parameters is straightforward by following the general coordinate ascent algorithm in [40]. For completeness, the detailed algorithm is reported in Section I (see the Supplementary Material).

3) Refine the Clusters: The clusters estimated from the VB algorithm are coarse in the sense that it only provides an approximate inference of DPM. In addition, DPM tends to produce many singleton clusters when the sample size is large. Hence, we need an algorithm to refine the clusters, which can be accomplished through merging similar clusters. An intuitive choice is to apply "clustering of clusters," which is able to merge a large number of clusters into a smaller number of larger clusters. In order to do so, we implement a recently proposed algorithm SIGN [41] for refining the cluster estimates and merging tiny clusters. It takes three sets of inputs: cluster assignments $\tilde{Y} = \{\tilde{y}_1, \ldots, \tilde{y}_n\}$ with $\tilde{y}_i \in \{1, \ldots, B\}$, cluster-specific parameters $\{\tilde{\mu}_b, \tilde{\Lambda}_b\}_{b=1}^{B}$, and cluster-specific summary statistics $\{\tilde{m}_b, \tilde{S}_b\}_{b=1}^{\tilde{B}}$, where $\tilde{m}_b$ and $\tilde{S}_b$ are the sample mean and covariance matrix of features $\{z_i | \tilde{y}_i = b\}$ in cluster b, respectively. In our context, $\tilde{Y}$ and $\{\tilde{\mu}_b, \tilde{\Lambda}_b\}_{b=1}^{B}$ are estimated from the VB algorithm and the summary statistics are easy to compute given $\tilde{Y}$. Through sampling from the approximate posterior distribution of the DPM model, this algorithm returns three sets of outputs: the refined/merged clusters $Y = \{y_1, \ldots, y_n\}$ with $y_i \in \{1, \ldots, K\}$, updated cluster-specific parameters $\{\mu_k, \Lambda_k\}_{k=1}^{K}$, and updated cluster-specific summary statistics $\{z_k, s_k\}_{k=1}^{K}$, which will be sent to our clustering loss for backpropagating the network. Therefore, with this step, we can make the number of clusters much smaller than that from VB, i.e., $K \ll B$. The details of the SIGN algorithm are provided in Section II (see the Supplementary Material).

4) Backpropagate the Networks: In our backward step, we need to train the deep network as well as extract the deep representations. A gradient descent method is usually used for updating the network parameter θ, for which we need to specify a single loss function with the pseudo labels Y and $\{\mu_k, \Lambda_k\}_{k=1}^{K}$. For coherency, we first derive a clustering loss from the log-posterior distribution of DPM

$$\begin{aligned} L_0(\theta, \{\pi_k, \mu_k, \Lambda_k\}_{k=1}^{\infty} | X) \\ = -\sum_{i=1}^{n} \log \sum_{k=1}^{\infty} \pi_k |\Lambda_k|^{1/2} \\ \times \exp\left(-\frac{1}{2}(f_\theta(x_i) - \mu_k)^T \Lambda_k (f_\theta(x_i) - \mu_k)\right). \end{aligned} \quad (5)$$

The optimization of (5) is infeasible due to the infinite sum. To resolve this issue, we integrate out π_k from (5), which yields a loss function without intractable terms

$$\begin{aligned} L_0(\theta, Y, \{\mu_k, \Lambda_k\}_{k=1}^{K} | X) \\ = \sum_{i=1}^{n} (f_\theta(x_i) - \mu_{y_i})^T \Lambda_{y_i} (f_\theta(x_i) - \mu_{y_i}). \end{aligned} \quad (6)$$

The loss function defined in (6) is calculated based on the clusters of samples, which should be estimated using all the data points. In order to use batch-based optimization for training the network, we generate the cluster labels and cluster-specific parameters at the beginning of each period (which is a similar concept to "epoch" in supervised learning), namely, the forward pass is taken on the entire dataset in every period. In the backward step, we can easily use backpropagation with mini-batch stochastic gradient descent (MGD) to update the network parameters θ given the generated pseudo labels Y and cluster-specific parameters (μ_k, Λ_k) in every training batch.

B. "Trivial Solution" Problem

"Trivial solution" is a common problem across many unsupervised learning methods, especially for the deep clustering approaches that jointly learn discriminative features and assign cluster labels to the extracted features [9]. When projecting the high-dimensional data onto low-dimensional space, the deep net $f_\theta(\cdot)$ tends to produce a collapsed feature space so that any clustering rule will fail to discriminate these features. As a consequence, most of the observations are grouped in very few clusters with a very low clustering loss value. For example, if the network parameters are all close to zero $\theta \approx 0$, then the embedded data vectors will be all around zeros as well $Z = f_\theta(X) \approx 0$ for any X. This gives rise to a single cluster with nearly optimal clustering loss. A common solution used by existing deep clustering methods is penalizing a minimal number of samples per cluster [9], [46]. Together with a fixed number of clusters, it avoids the problem of features being collapsed into few clusters. However, we are facing a more challenging problem due to the undetermined number of clusters. In this section, we will provide a simple effective solution inspired by the determinantal point processes [47], [48], and our solution can also be generalized to other unsupervised learning algorithms.

1) Repulsion and Regularization: Let Σ_Z denote the sample covariance matrix of the extracted features $Z = f_\theta(X)$. The determinant $\det(\Sigma_Z)$ can be interpreted as the volume of a parallelotope spanned by the column vectors of Σ_Z. Similar column vectors span less volume than distinct ones. In other words, small $\det(\Sigma_Z)$ leads to degenerated/attractive features, whereas large $\det(\Sigma_Z)$ leads to separable/repulsive features. The latter allows us to resolve the trivial solution issue. To avoid small $\det(\Sigma_Z)$, we minimize the loss function L_0 with an additional constraint

$$\min \; L_0, \quad \text{s.t. } \det(\Sigma_Z) > c. \quad (7)$$

2) Objective Function: The determinant $\det(\Sigma_Z)$ usually has a very large scale, which makes the constraint on (7) numerically unstable, so we convert it to a log-determinant optimization problem given that the sample covariance matrix Σ_Z is always positive semidefinite. In our experiments, the positive definiteness is guaranteed given that we choose a large batch size. If a small batch size is required, a matrix shrinkage can be applied by adding a small multiple of the identity matrix to Σ_Z, i.e., $\Sigma_Z + \lambda_0 I$, where λ_0 is an additional hyperparameter. By placing a constraint on the log determinant of Σ_Z, we have

$$\min \; L_0, \quad \text{s.t. } \log\det(\Sigma_Z) > c'. \quad (8)$$

Algorithm 1 Clustering Algorithm for Updating θ and Cluster Labels Y

Input:
X: a sequence of images
Output:
θ: network parameters; Y: image labels;
Z: deep representations
Initialize:
θ: CNN parameters; period $p = 1$
while *not converged* **do**
a. Generate $Z^{(p)}$ from $f_{\theta^{(p-1)}}(X)$
b. Update cluster labels $\tilde{y}^{(p)}$ and cluster-specific parameters $\tilde{\mu}_k^{(p)}$, $\tilde{\Lambda}_k^{(p)}$ using VB algorithm
c. Refine the clusters $y^{(p)}$, ${\mu_k}^{(p)}$, ${\Lambda_k}^{(p)}$
d. Update $\theta^{(p)}$ given $y^{(p)}$, ${\mu_k}^{(p)}$, ${\Lambda_k}^{(p)}$
for Iteration $t = 1, \cdots, T$ **do**
Update θ through MGD using loss from (9)
end for
$p \leftarrow p + 1$
end while

Letting $\mathbf{\Omega}_Z = \Sigma_Z^{-1}$, (7) is equivalent to the following optimization problem (Lagrange multipliers):

$$\min \; L_0 + \lambda_R \log\det \mathbf{\Omega}_Z \tag{9}$$

where λ_R is a balancing parameter. We will refer to the new regularization as "repulsion" and use L_R to denote it in the following discussion.

Through adding a regularization term to (6), we get our training objective function in (9). With this training objective, the optimization workflow is outlined in Algorithm 1. The final algorithm is quite straightforward. In each training period, we perform a forward pass, and for each training batch, we perform a backward pass to update the network. Generally, the algorithm is considered as being converged when the whole training objective cannot be improved further. In practice, we choose and save the checkpoint for the model with the minimum L_0^p from the periods p's that hit a relative tolerance on the "repulsion" within a certain number of periods, i.e., $(L_R^{(p)} - L_R^{(p-1)})/L_R^{(p-1)} \leq \epsilon$, where ϵ is a small relative tolerance value and is chosen to be 0.01 in our experiments. Our optimization flow is end-to-end and easy to be implemented. We use noninformative priors for the DPM model, so there is a minimal need to tune the hyperparameters in the proposed algorithm. In order to have a good starting point, we initialize the network in three steps. First, we flatten the image arrays to 1-D vectors and apply PCA to transform the vectors to low-dimensional space. Second, we run the VB algorithm on the transformed data to generate cluster labels from DPM. Finally, we append a classification layer to the deep network and update the network parameters by minimizing the cross-entropy loss until convergence, taking the cluster labels generated from VB as ground truth.

IV. Experiments

We demonstrate the utility of the proposed method in comparison with unsupervised learning methods that assume a known number of clusters on benchmark datasets.

TABLE I
Data Information

Dataset	#Samples	Image Size	#Classes
YTF	10,000	55×55	41
USPS	11,000	16×16	10
MNIST-test	10,000	28×28	10
UMist	575	112×92	20
FRGC	2462	32×32	20

A. Datasets

We evaluate our clustering performance on five benchmark image datasets that are commonly used for evaluating deep clustering methods. These include handwritten digits datasets, USPS[1] and MNIST-test [2], and face image datasets, UMist [49], FRGC-v2.02,[2] and Youtube-Face (YTF) [50]. USPS has 11 000 handwritten digit images from zero to nine. MNIST-test is a dataset that consists of 10 000 handwritten digit images. For FRGC-v2.0 and YTF datasets, we follow the preprocessing work in [5] and [8] and, respectively, choose the same 41 subjects and 20 subjects from the original datasets. The sample size, the image size, and the number of classes for all the datasets are shown in Table I.

B. Clustering Performance

The experimental setups and implementation details for running DNB can be found in Section III (see the Supplementary Material). We consider as baselines a couple of unsupervised learning methods that require a known number of clusters for comparison, including K-means [51], NJW spectral clustering (SC-NJW) [52], self-tuning spectral clustering (SC-ST) [53], large-scale spectral clustering (LS-SC) [54], agglomerative clustering with average linkage (AC-Link) [1], zeta-function based agglomerative clustering (AC-Zell) [55], normalized cuts (N-Cuts) [56], spectral embedded clustering (SEC) [57], local discriminant models and global integration (LDMGI) [58] and locality-preserving NMF (NMF-LP) [59], NMF with deep model (NMF-D) [60], deep embedded clustering (DEC) [61], and additionally two recent state-of-the-art deep clustering methods, joint unsupervised learning (JULE) [8], and deep embedded regularized clustering (DEPICT) [5] with a grid search for the unknown number of clusters. Specifically, we run the algorithms with a value from grid points of $\{5, 10, \ldots, 100\}$ as the number of clusters and then select the optimal number of clusters using a couple of common clustering validity indices following their corresponding selection rule [62]. The estimated clustering labels and embedding data are used as inputs to calculate these indices. The clustering validity indices we apply are: cubic clustering criterion (CCC) [30], silhouette score (SC) [26], Dunn index (DUNN) [27], Cindex (CIND) [29], Ptbiserial index (PTB) [31], [63], DB index (DB) [32], and SDBW index (SDBW) [33]. We report the results for JULE[3] and DEPICT[4] based on the optimal number of clusters determined with each index. Some

[1] https://cs.nyu.edu/roweis/data.html
[2] http://www3.nd.edu/cvrl/CVRL/Data_Sets.html
[3] https://github.com/jwyang/JULE.torch
[4] https://github.com/herandy/DEPICT

TABLE II

QUANTITATIVE EVALUATION OF DIFFERENT CLUSTERING ALGORITHMS ON THE BENCHMARK DATASETS FOR CLUSTERING PERFORMANCE (NMI AND ACC). THE MEANS AND STANDARD DEVIATIONS OF NMI AND ACC ARE REPORTED. THE DASH MARK (-) INDICATES THAT THE RESULT IS MISSING OR IMPRACTICAL TO OBTAIN

Dataset	YTF		USPS		MNIST-test		UMist		FRGC	
	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC
K-means	0.761	0.548	0.447	0.467	0.528	0.560	0.609	0.419	0.389	0.327
SC-NJW	0.752	0.551	0.690	0.413	0.755	0.220	0.727	0.551	0.186	0.178
SC-ST	0.620	0.290	0.726	0.308	0.756	0.454	0.611	0.411	0.431	0.358
SC-LS	0.759	0.544	0.681	0.659	0.756	0.740	0.810	0.568	0.550	0.407
N-Cuts	0.742	0.536	0.675	0.314	0.753	0.304	0.782	0.550	0.285	0.235
AC-Link	0.738	0.547	0.579	0.421	0.662	0.693	0.643	0.398	0.168	0.175
AC-Zell	0.733	0.519	0.799	0.575	0.768	0.693	0.755	0.517	0.351	0.266
SEC	-	-	0.511	0.544	0.790	0.815	-	0.633	-	-
LDMGI	-	-	0.563	0.580	0.811	0.847	0.866	0.691	-	-
NMF-LP	0.720	0.546	0.435	0.522	0.467	0.479	0.560	0.365	0.346	0.259
BNP	0.831 (0.008)	0.506 (0.012)	0.642 (0.005)	0.169 (0.015)	0.621 (0.003)	0.157 (0.005)	0.776 (0.006)	0.273 (0.018)	0.538 (0.017)	0.213 (0.003)
NMF-D	0.562	0.536	0.287	0.382	0.241	0.250	0.500	-	0.259	0.274
DEC	0.446	0.371	0.586	0.619	0.827	0.859	0.713	0.552	0.505	0.378
JULE + SC	0.911 (0.002)	0.683 (0.008)	0.927 (0.014)	0.944 (0.041)	0.916 (0.001)	0.912 (0.027)	0.840 (0.015)	0.486 (0.063)	0.669 (0.027)	0.543 (0.009)
DEPICT + SC	0.462 (0.004)	0.229 (0.000)	0.662 (0.058)	0.478 (0.024)	0.660 (0.219)	0.600 (0.313)	-	-	0.454 (0.172)	0.364 (0.108)
JULE + DUNN	0.790 (0.113)	0.547 (0.116)	0.743 (0.046)	0.464 (0.056)	0.767 (0.153)	0.600 (0.325)	0.829 (0.124)	0.605 (0.253)	0.627 (0.099)	0.523 (0.037)
DEPICT + DUNN	0.462 (0.004)	0.229 (0.000)	0.889 (0.026)	0.852 (0.055)	0.826 (0.04)	0.721 (0.139)	-	-	0.458 (0.191)	0.373 (0.127)
DNB	0.884 (0.008)	0.658 (0.008)	0.835 (0.020)	0.724 (0.031)	0.860 (0.004)	0.841 (0.025)	0.851 (0.042)	0.710 (0.056)	0.651 (0.021)	0.464 (0.020)

competing methods are reported with their best results from either their original papers or from previous literature [5], [8], [64]. When the result is reported as unavailable/impractical to obtain in this literature and/or the implementation is not publicly available, we put a dash mark (-) for this result (Table II and Table II, see the Supplementary Material). In addition, we include the results by directly running BNP on the principal components of the flattened image data.

We report the results of DNB and the baselines that are reimplemented by us averaged over three runs. We evaluate the clustering performance with two commonly used metrics, normalized mutual information (NMI) [5], [8], [65] and clustering accuracy (ACC). NMI measures the similarity of two different cluster assignments A and B. The NMIs and ACCs of all the methods are reported in Table II and the average results for our methods and baselines that are reimplemented by us are provided with the sample standard deviations included in the bracket. The training curves are shown in Fig. 1 (see the Supplementary Material), which include how the levels of total loss, clustering loss, L_0, repulsion regularization L_R, and clustering accuracy change over periods.

Even though our method does not need the information of the number of clusters that is required by the other methods as an input, DNB still outperforms most of the competing methods across different datasets. Also, note that DNB outperforms DEC, a popular autoencoder-based deep clustering method, in our comparison. We further present the clustering performance of JULE and DEPICT when they are combined with the various aforementioned clustering validity indices. We report the results with SC and DUNN in Table II and those with other indices in Table II (see the Supplementary Material). Overall, we find that DNB outperforms DEPICT with both of these two indices only except for DEPICT + DUNN on USPS. DNB has a better performance than JULE + DUNN but performs moderately worse than JULE + SC. Fig. 3 (see the Supplementary Material) shows the running time for DNB as well as JULE and DEPICT with grid search on K. DNB is significantly faster. We conclude that with an unknown number of clusters, running existing deep clustering methods with an arbitrary selection approach cannot guarantee a better performance than our method even at a much higher computational cost. Note that when the true number of clusters is unknown, which clustering validity index is "optimal" is also unknown. Thus, the choice of index itself is very challenging and has a significant effect on the clustering performance as we see from our experiments that the results of JULE and DEPICT have large variability across different clustering validity indices. Therefore, we believe that the proposed DNB is useful in practice as a new deep clustering method that does not require the users to input the number of clusters or to select an optimal clustering validity index. In summary, as a joint learning method that does not specify the number of clusters *a priori*, DNB obtains a promising clustering performance.

C. Sensitivity Analysis of DPM Hyperparameters

The most influential parameter of the DPM model is the DP concentration parameter α, which allows for the inference on the number of clusters from data. Parameter α is usually set with $\alpha = (\alpha^*/K^*)$, where α^* can be chosen in a noninformative way (e.g., in our experiments, $\alpha^* = 1$) so that the data quickly "washes out" the influence of hyperparameters as the sample size grows. To investigate the effect of α^*, in addition to $\alpha^* = 1$, we report the clustering performance on all datasets with $\alpha^* \in \{100, 200, 400\}$ in Fig. 2, which corresponds to $\{1, 2, 4\}$ for α given $K^* = 100$ in our experiments.

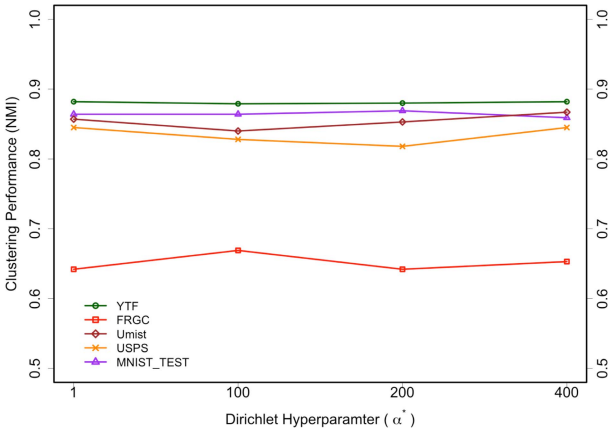


Fig. 2. Sensitivity analysis of Dirichlet hyperparameters.

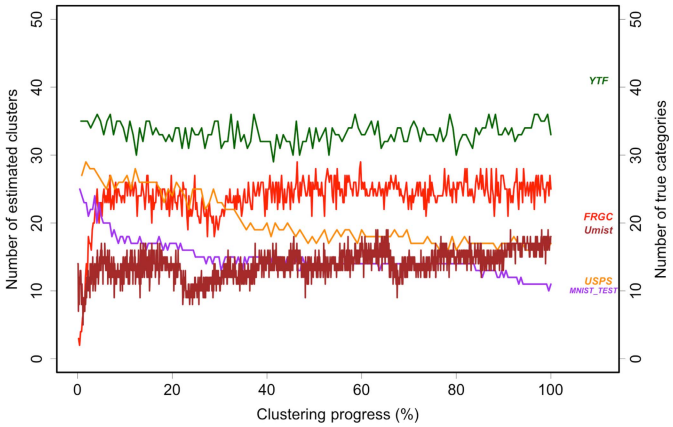


Fig. 3. Number of estimated clusters changing during the learning progress. The line and label for the same dataset are in the same color.

We also provide a sensitivity analysis of α^* over 11 evenly spaced values from [0.5, 5.0] in Fig. 2 (see the Supplementary Material) to show how α^* affects the performance in a finer range. We find that clustering performance is relatively robust to the choice of α^* within the tested ranges and the specific datasets under consideration, which suggests the effectiveness of applying a noninformative prior in our experiments. If desired, α can also be estimated by an empirical Bayes approach or assigned a gamma hyperprior. Although the latter still requires the specification of hyperparameters of gamma, in general, those hyperparameters will have a smaller influence on the clustering results as they are higher in the hierarchy of the hierarchical model. We remark that there are other factors that can influence the number of clusters K, such as the hyperparameters in the base measure G_0. Our recommendation is to set them to be noninformative and maximally let the data dictate the determination of K.

D. Finding the Number of Estimated Clusters

We further investigate the accuracy in estimating the number K of clusters. In Fig. 3, we report the number of estimated clusters changing with the learning process. We focus on the major clusters that contain $>1\%$ of the images because the BNP model is known to generate many tiny clusters that are often hard to interpret [66]. For the two handwritten digits datasets, the number of clusters approaches the truth as the algorithm processes. The number of clusters for the MNIST-test can be accurately detected, whereas the number for USPS is slightly overestimated. For the face image data, DNB still performs reasonably well: it accurately estimates K for UMist, slightly overestimates K for FRGC, and slightly underestimates K for YTF.

E. Ablation Experiments

In our experiments, we adopt a PCA-based pretraining initialization instead of random initialization to get a good starting point for the network weights. To investigate the effect of our initialization strategy and also check the robustness of our model against random initialization, we run our DNB algorithm with a random initialization and present the results comparing with those using the pretraining initialization in Table III. As shown in the first two rows for each dataset in Table III, our initialization achieves a slight improvement around 1%–4% in terms of NMI and similarly ACC for all the datasets except for YTF. DNB still obtains a good performance over all the datasets with the random initialization, which suggests the robustness of our model against random initialization as well as the benefit of using the designed initialization strategy.

The SIGN algorithm and DPM model are the two important components in our learning framework. To investigate the impact of the SIGN algorithm, we train our model by dropping this component. Also, to clarify the effectiveness of the DPM, we run our model by replacing the DPM with a finite mixture model, i.e., GMM, and choosing the number of clusters, K, for GMM with the same value of K^* from our DPM. We also drop SIGN and replace DPM with GMM at the same time to further evaluate their joint effects. As shown in Table III, the performances of dropping SIGN only and replacing DPM with GMM only are both reduced compared with that of the proposed full model for most of the datasets. We find that the performance decreases very slightly for FRGC, and the performance of replacing DPM with GMM does not change significantly for UMist, while the performance drops a lot when SIGN is disabled. However, when we drop SIGN and replace DPM with GMM together, the training of FRGC and UMist both fails due to exploded losses. In this case, the performance of other datasets except for YTF also drops to a larger extent compared to dropping or changing one of the two components. From these results, we conclude that the use of SIGN contributes to the clustering performance by refining the cluster estimates, but our model does not heavily depend on SIGN when the BNP clustering component, i.e., the DPM in our experiments, is retained. When SIGN is dropped, the different performance between DPM and GMM with a large fixed K among most of the datasets validates the importance of applying a BNP model for clustering when K is unknown. An interesting finding is that when SIGN is applied for refining clusters, using GMM with a large K can lead to a good performance on some datasets. It is likely because SIGN is based on a BNP model and plays a role in refining the cluster estimates from GMM. We additionally report a comparison of the number of resultant clusters between with and without

TABLE III

ABLATION STUDIES OF DNB ON INITIALIZATION, SIGN, DPM/GMM, AND REPULSION REGULARIZATION. THE MEANS AND STANDARD DEVIATIONS OF NMI AND ACC ARE REPORTED. AN X-MARK INDICATES FAILED TRAINING

Dataset	Pretrain	*SIGN*	DPM GMM	Rep	NMI/ACC
YTF					
	✓	✓	✓	✓	0.884/0.658 (0.008/0.008)
		✓	✓	✓	0.881/0.647 (0.001/0.004)
	✓		✓	✓	0.875/0.656 ((0.013/0.013))
	✓	✓		✓	0.865/0.583 (0.004/0.010)
	✓			✓	0.868/0.570 (0.002/0.010)
	✓	✓	✓		✗
USPS					
	✓	✓	✓	✓	0.835/0.724 (0.020/0.031)
		✓	✓	✓	0.790/0.674 (0.010/0.011)
	✓		✓	✓	0.818/0.755 (0.005/0.005)
	✓	✓		✓	0.702/0.299 (0.005/0.017)
	✓			✓	0.691/0.295 (0.018/0.062)
	✓	✓	✓		✗
MNIST-test					
	✓	✓	✓	✓	0.860/0.841 (0.004/0.025)
		✓	✓	✓	0.822/0.775 (0.008/0.028)
	✓		✓	✓	0.775/0.754 (0.016/0.005)
	✓	✓		✓	0.721/0.368 (0.004/0.006)
	✓			✓	0.718/0.366 (0.004/0.023)
	✓	✓	✓		✗
UMist					
	✓	✓	✓	✓	0.851/0.710 (0.042/0.056)
		✓	✓	✓	0.838/0.667 (0.048/0.067)
	✓		✓	✓	0.770, 0.503 (0.041, 0.044)
	✓	✓		✓	0.873/0.629 (0.009/0.012)
	✓			✓	✗
	✓	✓	✓		✗
FRGC					
	✓	✓	✓	✓	0.651/0.464 (0.021/0.020)
		✓	✓	✓	0.642/0.463 (0.012/0.007)
	✓		✓	✓	0.645/0.463 (0.018/0.024)
	✓	✓		✓	0.672/0.430 (0.014/0.013)
	✓			✓	✗
	✓	✓	✓		✗

SIGN refinement when DPM is retained in Table III (see the Supplementary Material). As shown in Table III (see the Supplementary Material), SIGN is able to significantly reduce the number of clusters through refining clusters.

Finally, we investigate the effect of the new regularization "Repulsion" (Rep) by setting $\lambda_R = 0$ in our objective training function [see (9)]. The last row for each dataset in Table III shows that dropping "Repulsion" will fail the training for all the datasets due to clusters being collapsed into one, which proves that the "Repulsion" regularization is critical to avoid the "trivial solution" in our training framework.

V. CONCLUSION

In this article, we consider a doubly unsupervised learning problem for image data. We present a deep BNP clustering framework, DNB, for jointly learning image clusters and deep representations without specifying the number of clusters. DNB trains the deep neural networks with the DPM model in a recurrent fashion, which consists of a forward step for generating cluster labels and a backward step for propagating the clustering loss through the network. We further integrate a regularization approach to address the problem of "trivial solution." We validate DNB through extensive experiments comparing with many existing clustering methods.

In this article, we focus on a general algorithm that addresses an unsolved problem for deep clustering with an unknown number of clusters. In order to perform an extensive comparison with many clustering methods, we demonstrate the clustering performance on several benchmark large-scale datasets and choose network architectures following the previous work [5], [8]. As being considered crucial for the performance of supervised learning, the architecture of networks also plays an important role in the performance of deep clustering. We did not specifically search the network architectures and simply adopted simple designs that are similar to what has been deployed in previous work [8], as designing new network architectures is not the focus of this work. Given it is not straightforward to apply the existing neural architecture search and hyperparameter tuning algorithms that have been developed for supervised learning, some work has been proposed to apply an ensemble paradigm to alleviate the impact of different architectures and hyperparameters [67], which can be our future research direction. On the other hand, as limited by the running time and computational resources such as the experiments performed in other deep clustering literature, at this point, we do not explore our algorithm on more noisy, difficult, and much larger datasets, e.g., ImageNet, which may take weeks for training [9] and require more delicate designs of network architecture and training strategy, and thus, we leave this extension for future work.

REFERENCES

[1] A. Jain, M. Murty, and P. Flynn, "Data clustering: A review," *ACM Comput. Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[2] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[3] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Proc. Int. Conf. Artif. Neural Netw.*, 2011, pp. 52–59.

[4] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 1096–1103.

[5] K. G. Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, "Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5736–5745.

[6] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3861–3870.

[7] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, "A survey of clustering with deep learning: From the perspective of network architecture," *IEEE Access*, vol. 6, pp. 39501–39514, 2018.

[8] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5147–5156.

[9] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 132–149.

[10] A.-A. Nahid and Y. Kong, "Involvement of machine learning for breast cancer image classification: A survey," *Comput. Math. Methods Med.*, vol. 2017, pp. 1–29, Dec. 2017.

[11] D. Steinberg, A. Friedman, O. Pizarro, and S. B. Williams, "A Bayesian nonparametric approach to clustering data from underwater robotic surveys," in *Proc. Int. Symp. Robot. Res.*, vol. 28, 2011, pp. 1–16.

[12] B. J. Ferdosi, H. Buddelmeijer, S. Trager, M. H. F. Wilkinson, and J. B. T. M. Roerdink, "Finding and visualizing relevant subspaces for clustering high-dimensional astronomical data using connected morphological operators," in *Proc. IEEE Symp. Vis. Analytics Sci. Technol.*, Oct. 2010, pp. 35–42.

[13] Z.-Q. Wang, J. L. Roux, and J. R. Hershey, "Alternative objective functions for deep clustering," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 686–690.

[14] D. Reynolds, "Gaussian mixture models," *Encyclopedia Biometrics*, vol. 741, pp. 827–832, Jul. 2015.

[15] M. Yamamoto and H. Hwang, "A general formulation of cluster analysis with dimension reduction and subspace separation," *Behaviormetrika*, vol. 41, no. 1, pp. 115–129, Jan. 2014.

[16] K. Allab, L. Labiod, and M. Nadif, "A semi-NMF-PCA unified framework for data clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 1, pp. 2–16, Jan. 2017.

[17] L. Labiod and M. Nadif, "Efficient regularized spectral data embedding," in *Proc. Adv. Data Anal. Classification*, 2020, pp. 1–21.

[18] K. Allab, L. Labiod, and M. Nadif, "Simultaneous spectral data embedding and clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 6396–6401, Dec. 2018.

[19] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, "Auto-encoder based data clustering," in *Proc. Iberoamerican Congr. Pattern Recognit.*, 2013, pp. 117–124.

[20] J. Wang and J. Jiang, "An unsupervised deep learning framework via integrated optimization of representation learning and GMM-based modeling," in *Proc. Asian Conf. Comput. Vis.* Springer, 2018, pp. 249–265.

[21] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," 2015, *arXiv:1511.05644*. [Online]. Available: http://arxiv.org/abs/1511.05644

[22] W. Harchaoui, P.-A. Mattei, and C. Bouveyron, "Deep adversarial Gaussian mixture auto-encoder for clustering," 2017. [Online]. Available: https://openreview.net/pdf?id=rJFpDxfFl

[23] J. Tobias Springenberg, "Unsupervised and semi-supervised learning with categorical generative adversarial networks," 2015, *arXiv:1511.06390*. [Online]. Available: http://arxiv.org/abs/1511.06390

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[26] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, Nov. 1987.

[27] J. C. Dunn, "Well-separated clusters and optimal fuzzy partitions," *J. Cybern.*, vol. 4, no. 1, pp. 95–104, Jan. 1974.

[28] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *J. Roy. Stat. Soc. B, Stat. Methodol.*, vol. 63, no. 2, pp. 411–423, 2001.

[29] L. J. Hubert and J. R. Levin, "A general statistical framework for assessing categorical clustering in free recall," *Psychol. Bull.*, vol. 83, no. 6, p. 1072, 1976.

[30] W. Sarle, "Sas technical report a-108, cubic clustering criterion," SAS Inst., Cary, NC, USA, Tech. Rep. A-108, 1983. [Online]. Available: https://support.sas.com/documentation/onlinedoc/v82/techreport_a108.pdf

[31] G. W. Milligan, "An examination of the effect of six types of error perturbation on fifteen clustering algorithms," *Psychometrika*, vol. 45, no. 3, pp. 325–342, Sep. 1980.

[32] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Trans. Pattern Anal. Mach. Intell.*, vols. PAMI-1, no. 2, pp. 224–227, Apr. 1979.

[33] M. Halkidi and M. Vazirgiannis, "Clustering validity assessment: Finding the optimal partitioning of a data set," in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2001, pp. 187–194.

[34] M. Ester *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, 1996, vol. 96, no. 34, pp. 226–231.

[35] N. L. Hjort, C. Holmes, P. Müller, and S. G. Walker, *Bayesian Nonparametrics* (Cambridge Series in Statistical and Probabilistic Mathematics). Cambridge, U.K.: Cambridge Univ. Press, 2010.

[36] T. S. Ferguson, "A Bayesian analysis of some nonparametric problems," *Ann. Statist.*, vol. 1, no. 2, pp. 209–230, Mar. 1973.

[37] C. E. Antoniak, "Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems," *Ann. Statist.*, vol. 2, no. 6, pp. 1152–1174, Nov. 1974.

[38] J. Pitman and M. Yor, "The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator," *Ann. Probab.*, vol. 25, no. 2, pp. 855–900, Apr. 1997.

[39] A. Lijoi, R. H. Mena, and I. Prünster, "Controlling the reinforcement in Bayesian non-parametric mixture models," *J. Roy. Stat. Soc. B, Stat. Methodol.*, vol. 69, no. 4, pp. 715–740, Sep. 2007.

[40] D. M. Blei and M. I. Jordan, "Variational inference for Dirichlet process mixtures," *Bayesian Anal.*, vol. 1, no. 1, pp. 121–143, Mar. 2006.

[41] Y. Ni, P. Müller, M. Diesendruck, S. Williamson, Y. Zhu, and Y. Ji, "Scalable Bayesian nonparametric clustering and classification," *J. Comput. Graph. Statist.*, vol. 29, no. 1, pp. 53–65, Jan. 2020.

[42] Y. Ni, Y. Ji, and P. Müller, "Consensus Monte Carlo for random subsets using shared anchors," *J. Comput. Graph. Statist.*, vol. 29, no. 4, pp. 703–771, 2020.

[43] W.-A. Lin, J.-C. Chen, C. D. Castillo, and R. Chellappa, "Deep density clustering of unconstrained faces," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8128–8137.

[44] J. D. McAuliffe, D. M. Blei, and M. I. Jordan, "Nonparametric empirical bayes for the Dirichlet process mixture model," *Statist. Comput.*, vol. 16, no. 1, pp. 5–14, Jan. 2006.

[45] M. D. Escobar and M. West, "Bayesian density estimation and inference using mixtures," *J. Amer. Stat. Assoc.*, vol. 90, no. 430, pp. 577–588, Jun. 1995.

[46] A. Joulin and F. Bach, "A convex relaxation for weakly supervised classifiers," 2012, *arXiv:1206.6413*. [Online]. Available: http://arxiv.org/abs/1206.6413

[47] A. Kulesza, "Determinantal point processes for machine learning," *Found. Trends Mach. Learn.*, vol. 5, nos. 2–3, pp. 123–286, 2012.

[48] Y. Xu, P. Müller, and D. Telesca, "Bayesian inference for latent biologic structure with determinantal point processes (DPP)," *Biometrics*, vol. 72, no. 3, pp. 955–964, Sep. 2016.

[49] D. B. Graham and N. M. Allinson, "Characterising virtual eigensignatures for general purpose face recognition," in *Face Recognition*. Springer, 1998, pp. 446–456.

[50] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2011, pp. 529–534.

[51] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probab.*, vol. 1, no. 14, 1967, pp. 281–297.

[52] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. Conf. Neural Inf. Process. Syst.*, 2002, pp. 849–856.

[53] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," in *Proc. Conf. Neural Inf. Process. Syst.*, 2005, pp. 1601–1608.

[54] X. Chen and D. Cai, "Large scale spectral clustering with landmark-based representation," in *Proc. AAAI*, 2011, vol. 5, no. 7, p. 14.

[55] D. Zhao and X. Tang, "Cyclizing clusters via zeta function of a graph," in *Proc. Conf. Neural Inf. Process. Syst.*, 2009, pp. 1953–1960.

[56] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[57] F. Nie, D. Xu, I. W. Tsang, and C. Zhang, "Spectral embedded clustering," in *Proc. IJCAI*. Citeseer, 2009, pp. 1181–1186.

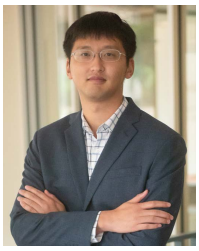
[58] Y. Yang, D. Xu, F. Nie, S. Yan, and Y. Zhuang, "Image clustering using local discriminant models and global integration," *IEEE Trans. Image Process.*, vol. 19, no. 10, pp. 2761–2773, Oct. 2010.

[59] J. Zheng, Y. Chen, Y. Jin, and W. Wang, "Incremental locality preserving nonnegative matrix factorization," in *Proc. 6th Int. Conf. Adv. Comput. Intell. (ICACI)*, vol. 9, Oct. 2013, pp. 1010–1015.

[60] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. Schuller, "A deep semi-NMF model for learning hidden representations," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1692–1700.
[61] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 478–487.
[62] M. Charrad, N. Ghazzali, V. Boiteau, and A. Niknafs, "Determining the number of clusters using NbClust package," in *Proc. MSDM*, 2014, p. 1.
[63] G. W. Milligan, "A Monte Carlo study of thirty internal criterion measures for cluster analysis," *Psychometrika*, vol. 46, no. 2, pp. 187–199, Jun. 1981.
[64] Z. Kang, X. Lu, J. Liang, K. Bai, and Z. Xu, "Relation-guided representation learning," *Neural Netw.*, vol. 131, pp. 93–102, Nov. 2020.
[65] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Proc. 26th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2003, pp. 267–273.
[66] P. Müller and R. Mitra, "Bayesian nonparametric inference–why and how," *Bayesian Anal.*, vol. 8, no. 2, pp. 269–302, Jun. 2013.
[67] S. Affeldt, L. Labiod, and M. Nadif, "Spectral clustering via ensemble deep autoencoder learning (SC-EDAE)," *Pattern Recognit.*, vol. 108, Dec. 2020, Art. no. 107522.
[68] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

Zeya Wang received the Ph.D. degree in statistics from Rice University, Houston, TX, USA, in 2017. He completed the post-doctoral training at the Department of Bioinformatics and Computational Biology, The University of Texas MD Anderson Cancer Center, Houston.

His research interests include applied mathematics, machine learning, biostatistics, and medical image analysis.



Yang Ni received the Ph.D. degree in statistics from Rice University, Houston, TX, USA, in 2015.

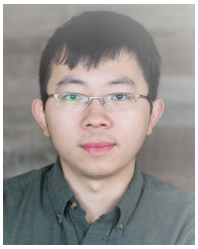
He was a Post-Doctoral Fellow with The University of Texas at Austin, Austin, TX. He is currently an Assistant Professor with the Department of Statistics, Texas A&M University, College Station, TX. His research interests include causal discovery, graphical models, Bayesian nonparametrics, big data computation, and machine learning.

Baoyu Jing received the bachelor's degree from Beihang University, Beijing, China, in 2016, and the master's degree from the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, in 2018. He is currently pursuing the Ph.D. degree with the Department of Computer Science, University of Illinois at Urbana–Champaign, Champaign, IL, USA.

His main research interests include data mining, graph mining, and transfer learning.

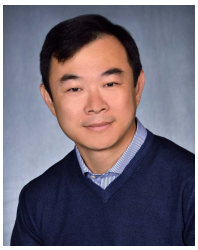
Deqing Wang received the Ph.D. degree in computer science from Beihang University, Beijing, China, in 2013.

He is currently an Associate Professor with the School of Computer Science and the Deputy Chief Engineer with the National Engineering Research Center for Science Technology Resources Sharing and Service, Beihang University. His research focuses on text categorization and data mining for software engineering and machine learning.



Hao Zhang received the Ph.D. degree from the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, in 2020.

He is currently a Post-Doctoral Scholar with the RISELab, University of California at Berkeley, Berkeley, CA, USA. His general research interest is in scalable machine learning, deep learning, and large-scale machine learning (ML) applications in computer vision and natural language processing.



Eric Xing (Fellow, IEEE) received the Ph.D. degree in molecular biology from Rutgers University, New Brunswick, NJ, USA, in 1999, and the Ph.D. degree in computer science from the University of California at Berkeley, Berkeley, CA, USA, in 2004.

He is currently a Professor of machine learning with the School of Computer Science and the Director of the CMU Center for Machine Learning and Health, Carnegie Mellon University, Pittsburgh, PA, USA. His principal research interests lie in the development of machine learning and statistical methodology, especially for solving problems involving automated learning, reasoning, and decision-making in high-dimensional, multimodal, and dynamic possible worlds in social and biological systems.

Dr. Xing is a member of the DARPA Information Science and Technology (ISAT) Advisory Group and the Program Chair of the International Conference on Machine Learning (ICML) 2014. He is also an Associate Editor of *The Annals of Applied Statistics* (AOAS), the *Journal of American Statistical Association* (JASA), the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (PAMI), and *PLOS Computational Biology* and an Action Editor of the *Machine Learning Journal* (MLJ) and the *Journal of Machine Learning Research* (JMLR).